IBM Cognos 8 Business Intelligence
Transformer

Version 8.4.1

Developer Guide

Information Management software

Product Information

This document applies to IBM Cognos 8 Version 8.4.1 and may also apply to subsequent releases. To check for newer versions of this document, visit the IBM Cognos Information Centers (http://publib.boulder.ibm.com/infocenter/cogic/v1r0m0/index.jsp).

Copyright

Table of Contents

Introduction

This document is intended for use with IBM Cognos Transformer, the OLAP modeling component delivered with IBM® Cognos® 8 Business Intelligence version 8.4 and subsequent releases.

You can use this *Developer Guide* as a reference when writing Model Definition Language (MDL) scripts for IBM Cognos Transformer.

Audience

This book is intended for experienced Transformer users who have an advanced understanding of Transformer concepts, functionality, and terminology. It provides all the information you need to understand MDL representation of Transformer models, but it does not explain the underlying Transformer features. Where possible, an equivalent in the user interface is given, and you should refer to the Transformer online Help or other Transformer documentation for more information.

Related Documentation

Our documentation includes user guides, getting started guides, new features guides, readmes, and other materials to meet the needs of our varied audience. The following documents contain related information and may be referred to in this document.

Note: For online users of this document, a Web page such as **The page cannot be found** may appear when clicking individual links in the following table. Documents are made available for your particular installation and translation configuration. If a link is unavailable, you can access the document on the IBM Cognos Resource Center (http://www.ibm.com/software/data/support/cognos_crc.html).

Documents	Description
IBM Cognos Connection User Guide	Using IBM Cognos Connection to publish, find, manage, organize, and view IBM Cognos content, such as scorecards, reports, analyses, and agents
IBM Cognos 8 Architecture and Deployment Guide	Understanding the IBM Cognos 8 architecture, developing installation strategies, including security considerations, and optimizing performance
IBM Cognos 8 Getting Started	Teaching new users how to use IBM Cognos 8
IBM Cognos 8 Installation and Configuration Guide	Installing, upgrading, configuring, and testing IBM Cognos 8, changing application servers, and setting up samples

Documents	Description
IBM Cognos 8 Transformer User Guide	Modeling and building PowerCubes using Model Definition Language (MDL) scripts
Framework Manager User Guide	Creating and publishing models using Framework Manager
Framework Manager Developer Guide	Creating and publishing models using the Framework Manager API
IBM Cognos 8 Administration and Security Guide	Managing servers, security, reports, and portal services; and setting up the samples, customizing the user interface and troubleshooting
Analysis Studio User Guide	Exploring, analyzing, and comparing dimensional data

Finding Information

Product documentation is available in online help from the **Help** menu or button in IBM Cognos products.

To find the most current product documentation, including all localized documentation and knowledge base materials, access the IBM Cognos Resource Center (http://www.ibm.com/software/data/support/cognos_crc.html).

You can also read PDF versions of the product readme files and installation guides directly from IBM Cognos product CDs.

Using Quick Tours

Quick tours are short online tutorials that illustrate key features in IBM Cognos product components. To view a quick tour, start IBM Cognos Connection and click the **Quick Tour** link in the lower-right corner of the Welcome page.

Getting Help

For more information about using this product or for technical assistance, visit the IBM Cognos Resource Center (http://www.ibm.com/software/data/support/cognos_crc.html). This site provides information on support, professional services, and education.

Printing Copyright Material

You can print selected pages, a section, or the whole book. You are granted a non-exclusive, non-transferable license to use, copy, and reproduce the copyright materials, in printed or electronic format, solely for the purpose of operating, maintaining, and providing internal training on IBM Cognos software.

Chapter 1: Model Definition Language (MDL)

MDL is the proprietary model definition language for Transformer. You use MDL to create, manipulate, update, and store models without accessing the Windows interface. You can set preferences globally and also automate Transformer functions for greater efficiency.

Tip: You may want to modify the cogtr.xml file, installed in the *installation_location*/configuration directory, to work more effectively in your particular production environment. For example, you can choose to work in structured MDL rather than verb format.

MDL represents Transformer models in text format (.mdl) and you can view the text in any text editor. The .py? file is a representation of the model, stored in default binary format. The question mark (?) in the extension .py? is replaced by the character used in your version of Transformer. Note that it is equivalent to the .mdl file, with the exception of any adjustments made on the Windows interface and, in some instances, passwords. For more information about Transformer file formats, see .py? and .mdl Formats.

MDL model statements can contain verbs, objects, and options. For example, in the statement
`DataSourceUpdate 257 "All Staff" SourceType ExcelDatabase`

- `DataSourceUpdate` is a verb specifying that a data source should be changed
- `257` is the object identifier and `All Staff` is its object name, identifying a specific data source object
- `SourceType ExcelDatabase` is the option that specifies the type of data source

You can use MDL in two ways:

- to create an MDL model that completely defines a Transformer model
- to create an MDL script that manipulates or updates an existing Transformer model

These uses can be combined. For example, you can append a script to the end of a model definition.

For examples of MDL scripts, see "Sample Scripts".

.py? and .mdl Formats

You can save Transformer models in two formats: .py? and .mdl. You can only open models saved as .py? files in the Transformer (Windows) interface. However, you can open models saved as .mdl files in either the Windows interface or a text editor.

Each .mdl file is a plain-text representation of the model, and may be compatible with other versions of Transformer than the one used to create it. However, it loads more slowly than a .py?-format file because it recreates all of the Transformer objects each time it loads.

You can open models saved as .mdl files on any supported UNIX, Linux, or Windows platform, making this the preferred format to use in a mixed production environment.

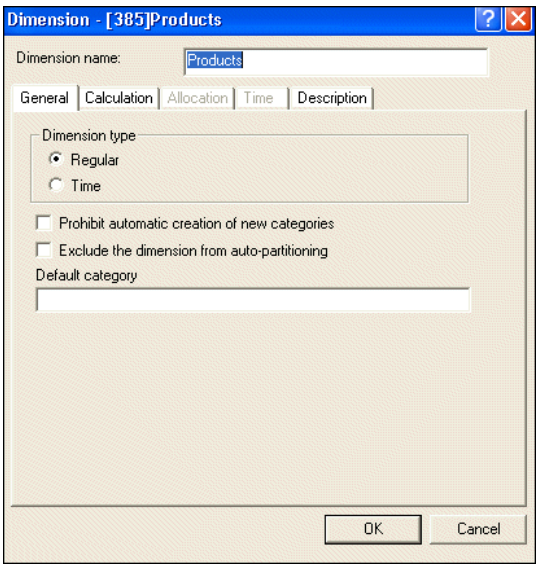
The .py? format is a binary representation of the model. Although not compatible between versions of Transformer, it loads faster than the .mdl-format equivalent, even though it is larger, because the Transformer objects are not recreated when the model reloads.

As you edit a model, the size of the associated .py? file increases because information about the operations performed during model editing is retained in the file. Transformer uses this information when performing incremental updates and other subsequent cube operations. Lengthy descriptions of the appended operations can cause fragmentation of the binary file. We therefore recommend that you periodically save your .py? models as .mdl files.

User Interface and MDL Representations of a Model Object

Based on an older version of the Great Outdoors sample, this illustration shows the **Products** dimension property sheet as it appears on the Transformer Windows interface.

To replicate this view, open the model on the Windows interface and, on the **Titles** tab of the **Preferences** dialog box, select the **Object identifier** display option. The code for the dimension now appears in the property sheet title; in our example, **Products** is identified by code 385.



To see the corresponding MDL definition, save the model as an .mdl file, and reopen it in a text editor. You can then use the **Find** utility with the identifier (385) to locate the MDL statement that defines that dimension.

Tip: We recommend viewing your text format .mdl files in an editor that enables line-by-line numbering because Transformer error messages refer to the line on which an error occurred. However, you can use any text editor to edit MDL, as long as you save the file in text format.

On the Windows interface, the dimension type is **Regular** and the **Prohibit automatic creation of new categories** option is not selected. These choices are represented in the MDL statement by the options `DimType Regular` and `NewCatsLock False` as follows:

```
Dimension 385 "Products" DimType Regular NewCatsLock
False ExcludeAutoPartitioning False DimDefaultCategory 0
```

Note: For easier readability, line breaks are inserted in lines over 70 characters in length. This does not affect the functionality.

MDL Usage

Although most users find it easier to create and maintain their Transformer models and cubes from the Windows interface, there are situations where an experienced Transformer modeler will benefit from using MDL.

The following are some of the ways you can use MDL, beginning with basic operations and progressing to more advanced applications:

- You can save your Transformer model as an .mdl file by selecting **Save As** from the **File** menu, changing the extension of the file to .mdl, and then saving the file. You can then use the .mdl file to open the model in subsequent versions of Transformer, move models between various production servers and platforms, or archive Transformer models.

  Models stored as .mdl files are compact, and because they can be opened in subsequent versions of Transformer, you need not worry about updating them.

  You can also use an .mdl file to improve the performance of Transformer models.

  Saving a .py? file in .mdl format integrates all of the changes made since the last MDL save into a clean model. It also prevents possible corruption of the model.

- You can open as an .mdl file in any text editor to perform global edits, such as changing all directory paths from C:\ to D:\ or changing all instances of "Years" to "Yrs".

- You can write scripts in MDL to manipulate Transformer models.

  For example, you can store repetitive tasks in an MDL batch file that can be run any time.

- You can use a text editor to work directly in .mdl files with Transformer models.

  For example, you can change a model on a UNIX server, create models as needed, and store parts or all of the resulting models in separate files.

- You can create custom Transformer applications.

Because MDL is an interpreted language, you can process .mdl files in any of the following ways:

- Open the .mdl file on the Transformer Windows interface.
- Run the .mdl file from the Windows command line using cogtr.exe.
- Access the .mdl file on a UNIX server, using the command line option `-m`.

Example - Changing a Model Object Using MDL

There are three ways that you can use MDL to modify an object in a Transformer model:

- Update the definition with a script.
- Change the MDL definition directly.
- Update the definition within the model.

For the sample model Great Outdoors.mdl, suppose you want to change the source file for Products (CSV) from Prodinfo.csv to P0001.csv. The original definition of the data source is as follows:

```
DataSourceMake 103 "Products (CSV)" Separator "," SourceType FlatFile_
ColNames CharacterSet Multibyte DecimalSep "."
Thousandsep "," Columns True Timing
PopYesCreateNo

Source ".\samples\powerplay\cubes and reports\prodinfo.csv" Speed False

SetCurrent False ServerSource False Presummarized False EnableMultiProcess
False
```

You can update the definition with a script that changes the data source file to P0001.csv. This script opens the model, issues a `DataSourceUpdate` statement, and saves the model.

```
OpenMDL "Great Outdoors.mdl"

DataSourceUpdate "Products (CSV)" Source "c:\Production\P0001.csv"

SaveMDL "Great Outdoors (Admin).mdl"
```

Alternatively, you can change the object definition in the model. In this case, you can delete the word `Prodinfo` in the `DataSourceMake` statement, and then replace it with `P0001`.

This is the new `DataSourceMake` statement:

```
DataSourceMake 103 "Products (CSV)" Separator "," SourceType FlatFile_ColNames
CharacterSet Multibyte DecimalSep "." Thousandsep "," Columns True Timing
PopYesCreateNo

Source "c:\Production\P0001.csv" Speed False

SetCurrent False ServerSource False Presummarized False
EnableMultiProcess False
```

You can also open the MDL model in a text editor and add this `DataSourceUpdate` statement to the end of the model definition file:

```
DataSourceUpdate "Products (CSV)" Source "c:\Production\P0001.csv"
```

The next time you open the model in Transformer, the change will be implemented. The original `DataSourceMake` statement is not altered, but your addition remains in effect as long as the `DataSourceUpdate` statement remains attached to the model definition.

If you update an object within the model, Transformer may be unable to complete the forward-referencing that is required in the definition of some objects. Therefore, you should use the verb `UpdateForwardReference` before the statement that updates the object, as follows:

```
UpdateForwardReference

DataSourceUpdate "Products (CSV)" Source "c:\Production\P0001.csv"
```

For information about command line options that you can use when running scripts, see the Transformer *User Guide*. For sample MDL scripts, see "Sample Scripts".

Recommendation - Set Transformer to Verb MDL

There are two types of Model Definition Language: verb MDL and structured MDL. Both types can be used to completely define a Transformer model. However, by default, Transformer generates models in structured format, the original version of MDL. This helps to ensure compatibility between models created in different versions of Transformer.

When you want to make changes to an existing model, we recommend that you use verb MDL, but first, you must add the following line to the cogtr.xml file:

```
<Preference Name="VerbOutput" Value="1"/>
```

Setting this parameter only affects the format of model definitions subsequently generated and saved by Transformer. Regardless of this setting, you can still use both verb and structured format on any output file.

For more information about using structured MDL, see "Structured MDL".

Chapter 2: Transformer Objects

This section describes the objects required to build models in Transformer and explains how to specify unique object identifiers, names, and locations in Model Definition Language (MDL).

An object is a definition of a particular component of the model. For example, the **Locations** data source object contains all the information in Transformer about the data source called **Locations**.

On the Windows interface, the **Locations** object is defined in the **Data Source** property sheet for **Locations**. In MDL, the same information is found in the MDL statement that defines the **Locations** data source object. The property sheet settings are represented in the MDL statement as options.

Each Transformer model must have at least one model object. The model may also contain one or more of each of the following object types:

Model object types	
Data Source	PowerCube
IBM Cognos Source (an IBM Cognos 8 package)	PowerCubeGroup
Column	PowerCubeGroupCube
Dimension	Subdimension
Level	Signon
Root Category	View
Drill Category	Custom View
Regular Category	Currency Table
Special Category	Currency
Measure	Dimension Calculation Definition

All Transformer objects have property sheets on the Transformer Windows interface. Only objects have property sheets.

Object Identifiers

You can track every Transformer object on the Windows interface and in MDL by its object identifier or its object name.

To show both object identifiers and object names on the Transformer Windows interface, ensure that both the **Object Identifier** and **Object Name** check boxes are selected on the **Titles** tab of the **Preferences** dialog box.

To exclude the object identifier from the Windows interface, clear the **Object Identifier** check box. You can also prevent the object identifier from appearing in the MDL file by clearing the **Save Object Unique Identifier values in MDL** check box on the **General** tab of the **Preferences** property sheet.

In addition to the requirement that the object identifier be unique within each model, Transformer object identifiers have several other defining characteristics, as follows:

- When auto-generated by Transformer (Windows), odd numbers identify all objects. This is a carry-over from Transformer version 7.x, which supported synchronized client-server operations, and assigned only even numbers to the equivalent server objects.
- Object identifiers are greater than 100 and less than 4,294,967,295.
- Once assigned, an object identifier cannot be changed by the modeler or the application.

As the modeler, you can specify a particular identifier and name when you create an object. For example, you can use the MDL statement `DataSourceMake 103 "Natsmall"` to uniquely identify the data source object by the identifier `103` and the name `Natsmall`.

However, manually assigned identifiers can be difficult to maintain, and cannot be used with incrementally updated cubes.

We therefore recommend that you allow Transformer to automatically generate the object identifiers. A different odd number is associated with each object, and an object name is assigned if it can be determined from the source data.

Tip: If you copy and paste parts of your model from different MDL files, we recommend that you change the `ObjecytIdOutput` setting in the cogtr.xml file to zero, to prevent object identifiers from appearing in the MDL files. Otherwise, you cannot guarantee that the object identifiers will remain unique after they are merged.

Object Names

If you create a Transformer object manually using MDL, you must supply a name for it.

Because object names are not necessarily unique within a model, if you identify objects only by name you often must supply more information to uniquely identify the object.

You can change object names, but not object identifiers. For example, one way to change the object name is to type the following:

```
DataSourceUpdate 103 "newname"
```

An object name must have the following structure:

- The name must be enclosed by single or double quotes.
- The name cannot contain a carriage return.
- The name cannot be more than 256 characters in length.

Uniqueness of Object Names

In certain contexts, object names for the same object type must be unique. These are the restrictions on the uniqueness of object names:

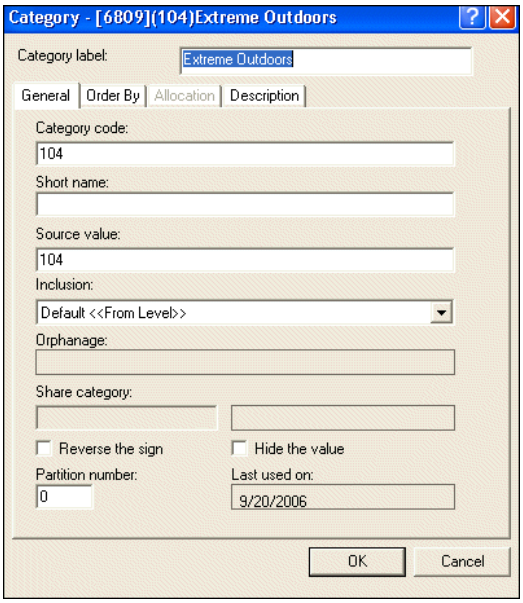
Object names	Restrictions on uniqueness
Category names	Must be within a dimension; applies to regular, root, drill, and special categories; see (p. 23).
Column names	Must be within a data source; see (p. 24).
Currency names	Must be within a model
Currency table names	Must be within a model
Data Source names, including package and report names	Must be within a model
DimCalcDef names	Must be within a dimension
Dimension names	Must be within a model
Level names	Must be within dimensions, if used in an advanced subset
Measure names	Must be within a model
PowerCube names	Must be within a model; cannot be the same as a PowerCube-Group name
PowerCubeGroup names	Must be within a model; cannot be the same as a PowerCube name
PowerCubeGroupCube names	Must be within a group
Signon names (IQD sources)	Must be within a model
Subdimension names	Must be within a dimension
View names	Must be within a dimension; see (p. 24).

Category Names

Categories are identified by a category label and a category code, rather than by a single name. On the Transformer Windows interface, category names are derived from category labels. However, in MDL, category names are derived from category codes.

Category codes are not the same as object identifiers. To show both on the Transformer Windows interface, you must select the **Object Identifier** and **Category Code** check boxes on the **Titles** tab of the **Preferences** dialog box.

For example, in this **Category** property sheet, both the object identifier **6609** and the category code **104** appear in the title bar for the category named **Extreme Outdoors**. The name is derived from the category label and cannot contain either a tilde (~) or an at sign (@).



In the corresponding MDL statement, the object identifier is the same: **6809**. However, the object name **104** is enclosed in double quotation marks and is based on the category code.

```
Category 6809 "104"
Parent 6425 Levels 1067 OrderBy Drill 1057 Value "104" Label "Extreme Outdoors"
Lastuse 20060920 SourceValue "104" Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False
```

Column Names

Column names are identical in MDL and on the Transformer Windows interface, unless they are manually changed. Column names cannot contain an at sign (@).

View Names

In structured MDL, views are represented by the following syntax:

```
ViewName id "Name"
```

where *id* is the unique Transformer identifier and "*Name*" is the name of the view.

Locating Objects Uniquely

If an object cannot be uniquely identified by its name or identifier, you must specify its relationship within a hierarchy of objects, such as its ancestors or level in a drill-down path.

When you use both an object identifier and an object name to identify an object in MDL, the object identifier determines whether the object exists. If there is a conflict between the object name and object identifier, the object identifier is used to locate the object, and the name is then changed accordingly.

If the object does not already exist or is not uniquely identified by its object name or object identifier, then you must supply the additional information required to locate it, in the syntax for each verb.

For example, suppose you want to refer to a column that exists in your data source. If your model does not provide a unique object name or object identifier for that column, you must uniquely identify it by specifying the data source in which it exists.

Similarly, if you add a new category to a dimension, in the context of a drill-down path, a level, and a parent category, you must specify all of those objects to locate it precisely and uniquely.

On the other hand, when you refer to an existing category, even if you do not uniquely identify it by its object identifier, you only need to specify the dimension name. This is sufficient because category names are unique within dimensions, and dimension names are unique within models.

The following rules govern the identification and location of objects:

- If the object already exists and is identified by its object identifier, then no other information is needed to locate the object in the model.
- If the object already exists and is identified only by its object name, then Transformer searches the model for an object with the same name and object type and uses the first one it finds.
- If the object does not already exist, then there must be sufficient contextual information to locate it in the model precisely.

Uniquely Locate Existing Objects

Use the following table to uniquely locate an existing object not already identified by its object identifier:

Object	Objects that must be referenced
Currency, Dimension, Measure, Model, Data Source (query), IBM Cognos Source (package or report), PowerCube, PowerCube Group, Signon	None
Category	Dimension
Column	Data source
DimCalcDef	Dimension
Drill Category	Dimension, Root Category
Level	Dimension, Drill Category

Object	Objects that must be referenced
PowerCube Group Cube	PowerCube Group, Dimension, Category
Root Category	Dimension
Subdimension	Dimension, Drill Category, Level, Parent Category
Special Category	Dimension, Parent Category
View	Dimension

Uniquely Locate New Objects

Use the following table to uniquely locate a new object:

Object	Objects that must be referenced
Currency, Dimension, Measure, Model, Data Source (query), IBM Cognos Source (package or report), PowerCube, PowerCube Group, Signon	None
Category	Dimension, Drill Category, Level, Parent Category
Column	Data source
DimCalcDef	Dimension
Drill Category	Dimension, Root Category
Level	Dimension, Drill Category, Parent Category
PowerCube Group Cube	PowerCube Group, Dimension, Category
Root Category	Dimension
Subdimension	Dimension, Drill Category, Level, Parent Category
Special Category	Dimension, Parent Category
View	Dimension

Setting Global Preferences

You can use the first few lines of your MDL script or a special MDL header file to specify global environment variables and processing preferences.

Specify a Preferred Category Sorting Option

To globally set a sorting option for all categories in the model, specify one of the following `ModelCategoryOrderDefault` options in the first few header lines of your MDL script:

- `OrderYes`
- `OrderNo`
- `OrderUsePreference`

For more information about each option, see the Transformer *User Guide*.

Chapter 3: Creating a Complete MDL Model

The following examples illustrate how to create complete models using Model Definition Language (MDL). Even if you never attempt this task yourself, by reviewing these examples you will learn the structure and interdependencies of Transformer MDL models.

Example - MDL Model Using IBM Cognos 8 Data Sources in Verb MDL

This example creates two IBM Cognos 8 data sources to be used in the model. The example shows the data source definitions, not the complete model.

Note: The `Add` form illustrates the verb that you use to define an object, to differentiate the syntax from that which Transformer uses when defining objects (the `Make` form). However, either form is acceptable. Also, you will note that each statement starts with the verb followed by a lengthy series of options.

When you create the IBM Cognos 8 data sources, you define the time and products dimensions and the measures used in the model.

Create the Data Sources in Transformer

This example is based on an IBM Cognos 8 package data source with one query and an IBM Cognos 8 report data source with one query.

```
CognosPackageAdd 103 "GO Sales
and Retailers" SourceType Package SourcePath "/content/package[@name='GOSales
and Retailers']" PackageTimeStamp "/content/package[@name='GOSalesand
Retailers']/model[@name='model']"

CognosPackageAdd 115 "GSR_rpt_1"SourceType
Report
SourcePath "/content/package[@name='GO Sales and Retailers']/report[@name='GSR_
rpt_1']"
PackageTimeStamp "/content/package[@name='GOSales and Retailers']/report
[@name='GSR_rpt_1']"

DataSourceAdd 105 "GO Salesand
Retailers~1" Separator "," SourceType CognosSourceQuery CharacterSetDefault
DecimalSep "." Thousandsep "," Columns True Timing
PopYesCreateDefaultPackageReportSource
103 "GO Sales and Retailers" AutoSummary TrueSetCurrent True  ServerSource
False Speed False Presummarized False

ColumnAdd 107 "[gosales_goretailers].[Orders].[Ordernumber]"
DataSource 105 Origin Source Offset 0 Column "Order number"Storage
Float64 Scale 0 Size 4 Decimals 0 InputScale 0 TimeArrayOff Rollup
CountAll

ColumnAdd 109 "[gosales_goretailers].[Orders].[Retailername]"
DataSource
105 Origin Source Offset 1 Column "Retailer name"Storage Text Scale
0
Size 102 Decimals 0 Class Description InputScale0 TimeArray Off
```

```
ColumnAdd 111 "[gosales_goretailers].[Orders].[Orderyear]"
DataSource
105 Origin Source Offset 2 Column "Order year"Storage Float64 Scale
0
Size 4 Decimals 0 InputScale 0 TimeArrayOff Rollup CountAll

ColumnAdd 113 "[gosales_goretailers].[Orders].[Ordermonth]"
DataSource
105 Origin Source Offset 3 Column "Order month"Storage Float64 Scale
0
Size 4 Decimals 0 InputScale 0 TimeArrayOff Rollup CountAll

DataSourceAdd 117 "GSR_rpt_1~1" Separator ","  SourceTypeCognosSourceQuery
CharacterSet Default DecimalSep "."  Thousandsep"," Columns True
Timing PopYesCreateDefaultPackageReportSource 115"GSR_rpt_1" AutoSummary
False SetCurrent True ServerSource False Speed False Presummarized
False

ColumnAdd 119 "[Report].[Query1xx.0].[Order number]"DataSource
117 Origin Source Offset 0 Column "Order number" StorageFloat64
Scale 0  Size 1 Decimals 0 Class Quantity InputScale 0 TimeArrayOff

ColumnAdd 121 "[Report].[Query1xx.0].[Order date]" DataSource117
Origin
Source Offset 1 Column "Order date" Storage Int32 Scale0 Size 1
Decimals 0 Class Date InputScale 0 TimeArray Off

ColumnAdd 123 "[Report].[Query1xx.0].[Product name]"DataSource
117 Origin Source Offset 2 Column "Product name" StorageText Scale
0  Size 1 Decimals 0 Class Description InputScale 0 TimeArrayOff

ColumnAdd 125 "[Report].[Query1xx.0].[Quantity]" DataSource117
Origin
Source Offset 3 Column "Quantity" Storage Float64 Scale 0 Size 1
Decimals
0 Class Quantity InputScale 0 TimeArray Off

ColumnAdd 127 "[Report].[Query1xx.0].[Revenue]" DataSource117
Origin
Source Offset 4 Column "Revenue" Storage Float64 Scale2  Size 1
Decimals 2 Class QuantityInputScale 0 TimeArray Off
```

Example - MDL Model Using an IQD Data Source in Verb MDL

In this example, separate files are created for each part of the model. The files are then merged by means of an MDL script, using the appropriate commands and their mandatory options to create the final model. This example does not include dimension views or custom views.

Note: The `Add` form illustrates the verb that you use to define an object, to differentiate the syntax from that which Transformer uses when defining objects (the `Make` form). However, either form is acceptable. Also, you will note that each statement starts with the verb, followed by a lengthy series of options. For readability, you may wish to insert carriage returns between the keywords.

This is the order in which you would create the separate files for a typical complete model based on an IQD data source, using verb MDL:

- Create the data sources in Transformer.
- Create the Time dimension.
- Create the Products dimension.
- Create key product measures.

- Create allocations.
- Create the signon.
- Populate the model and create the PowerCube.
- Create the model by combining the scripts.

Step 1: Create the Data Sources in Transformer

This example is based on anImpromptu Query Definition (IQD) data source. As a result, the initial `DataSourceAdd` and `ColumnAdd` MDL statements are copied directly from the Impromptu SQL string that is automatically generated when Transformer connects to the IQD data source.

Que_Col.mdl File

The first file adds information to the model from the relevant columns of the IQD data source.

```
DataSourceAdd "Products (CSV)" Separator "," SourceTypeFlatFile_ColNames
Source
"c:\prodinfo.csv" ColumnAdd "Product Line"DataSource "Products (CSV)"
Origin Generated Offset 0
Column "ProductLine" ColumnAdd "Product Type" DataSource "Products
(CSV)" Origin Generated
Offset 1
Column "Product Type" ColumnAdd "Product Id" DataSource "Products
(CSV)" Origin Generated
Offset 2
Column "Product Id" ColumnAdd "Product Name" DataSource "Products
(CSV)" Origin Generated
Offset 3
Column "Product Name" DataSourceAdd "Prod Line Plan" Separator ","
SourceType FlatFile_ColNames Source "c:\ProdPlan.asc"
ColumnAdd "YEAR" DataSource "Prod Line Plan" Origin Generated Offset 0
Column "Time"
Format Y DateLevel Year Class Date ColumnAdd "PROD_
LINE" DataSource "Prod Line Plan"Origin Generated
Offset 1
Column "Product Line"
ColumnAdd "FORECAST" DataSource "Prod Line Plan" Origin Generated
Offset 2
Column "Planned sales" DataSourceAdd "MAIN (IQD)" Separator","
SourceType DataSource Source "c:\bsc_msrs.iqd"
SQL 'select T1."ORDER_DT" as c1,
   T2."PROD_NO" as c2,
   T1."REP_NO" as '
'c3,
   T1."CUST_NO" as c4,
   T2."QTY" as c5,
   (T2."QTY" * T2."PRI'
'CE") as c6,
   T2."QTY" * T3."PROD_COST") as c7,
   CASE WHEN ((((T'
'2."QTY" * T2."PRICE") - (T2."QTY" * T3."PROD_COST"))/(T2."QTY" * T2."'
'PRICE")) <= 0.19) THEN (' "'Under
20%') WHEN ((((T2." '"QTY" * T2."PRICE")
- (T2."QTY" * T3."PROD_COST")) / (T2."QTY"* T2."PR'
'ICE"))  BETWEEN 0.2 AND 0.65) THEN (' "'20% - 65%') WHEN ((((T2." '"QTY" *
T2."PRICE"  -
(T2."QTY" * T3."PROD_COST")) / (T2."QTY" * T2."PR'
'ICE")) >= 0.66) THEN (' "'Over 65%') ELSE ('ERROR') END) as c8
from "
```

```
'"ORDER" T1,
 ("PRODUCT" T3 left outer join "ORDRDETL" T2 on T2."PROD_NO'
'" = T3."PROD_NO")
where (T2."ORDER_NO" = T1."ORDER_NO")
order by c1 asc'
",c3 asc,c4 asc,c2 asc"
ColumnAdd "Order Dt" DataSource "MAIN
(IQD)" Origin Source Offset 0 Column
"Time"
Storage Int32 Scale 0 Size 4 Decimals 0 Class Date InputScale 0 TimeArray Off
ColumnAdd "od-Prod No" DataSource "MAIN (IQD)"  Origin Source Offset 1 Column
"Product
Id"
Storage Float64 Scale 0 Size 5 Decimals 0 Class Quantity InputScale 0 TimeArray
Off
ColumnAdd "Revenue" DataSource "MAIN (IQD)" Origin Generated Offset
5 Column "Revenue" ColumnAdd "Cost" DataSource "MAIN
(IQD)" Origin Source Offset 6 Column "Cost"
Storage Float64 Scale 0 Size 6 Decimals 0
Class Quantity InputScale 0 TimeArray
Off
```

Step 2: Create the Time Dimension

When you create any dimension using MDL, you must manually create both the root category and drill category for that dimension.

The verbs `RootCatMake` and `DrillCatMake` are used in this example because there is no `Add` verb form for these objects. Also, you must set the inclusion property to `Suppress` for the drill category so that it does not appear in the reporting components.

Note: If you do not manually create the two default dimension views required for each dimension (All Categories and Omit Dimension), you must use the `ModelEnsureCompleteness` script to have Transformer create them automatically.

The definition of each level contains the specification for its dimension, drill category, and parent. Together, these define the hierarchical structure for the dimension.

Note: When you create a query in Transformer based on a Framework Manager package that contains hierarchical time-related categories, Transformer interprets the time-related categories as a regular dimension and not as a time dimension. As a result, the time dimension in your PowerCube will not contain any relative time categories. Ensure that you import all of the data needed to define your time dimension, and then create the date levels and categories.

Dim_Years.mdl FIle

The second file creates a standard Year-Quarter-Month time dimension for the model.

```
DimAdd "Years" DimType DateRootCatMake "Time" Dimension
"Years"DrillCatMake "By Time" Dimension "Years" Root "Time" Inclusion
Suppress Level Add "Year" Dimension "Years" Drill "By Time" Parent
""Source "Time"  Date Function Year Unique Categories True OrderBy
Drill "By Time" Column "Time" SortOrder Default SortAs Ascending
 LevelAdd "Quarter" Dimension "Years" Drill "By Time" Parent
"Year"  Source "Time" Date Function Quarter CatLabFormat 'YYYY
"Q"
Q'UniqueCategories True OrderBy Drill "By Time" Column "Time" SortOrder
Default
SortAs Ascending LevelAdd "Month"  Dimension Years" Drill "ByTime"
Parent
"Quarter"Source "Time" DateFunction Month CatLabFormat "YYYY/MMM"
```

```
UniqueCategories
True OrderBy Drill "ByTime" Column "Time"SortOrder Default  SortAs
Ascending
```

Step 3: Create the Products Dimension

The next step is to specify the details necessary to properly structure the primary dimension, Products.

Dim_Products.mdl

The third file pulls information needed for the Products dimension from the imported data.

```
DimAdd "Products" DimType Regular
RootCatMake "Product Line"
Dimension "Products"
DrillCatMake "By Product Line"
Dimension "Products"
Root "Product Line" Inclusion Suppress
LevelAdd "Product Line"
Dimension "Products"
Drill "By Product Line" Source "Product Line"
Associations "Product Line" AssociationType Type_Query
AssociationRole Role_Source
AssociationReferenced "Product Line"
LevelAdd "Product Type"
Dimension "Products"
Drill "By Product Line" Parent "Product Line" Source "Product Type"
Associations "Product Type" AssociationType Type_Query
AssociationRole Role_Source AssociationReferenced "Product Type"
LevelAdd "Product Id"
Dimension "Products"
Drill "By Product Line" Parent "Product Type" Source "Product Id"
Label "Product Name" UniqueCategories
True Associations "Product Id" AssociationType Type_Query
AssociationRole Role_Source AssociationReferenced "Product Id"
```

Step 4: Create Key Product Measures

Next, you need separate script files to add the three measures associated with the Products dimension.

Me_Prodcost.mdl

This file creates the Product Cost measure, which is taken from the Cost column.

```
MeasureAdd "Product Cost"
Associations "Cost" AssociationType Type_Query
AssociationRole Role_Source AssociationReferenced "Cost"
```

Me_Prodplan.mdl

This file creates the Product Plan measure, which is taken from the Planned Sales column.

```
MeasureAdd "Product Plan"
Associations "Planned Sales" AssociationType Type_Query
AssociationRole Role_Source
AssociationReferenced "Planned Sales"
```

Me_Revenue.mdl

This script creates the Revenue measure, which is taken from the Revenue column.

```
MeasureAdd "Revenue"
```

```
Associations "Revenue" AssociationType Type_Query
AssociationRole Role_Source
AssociationReferenced "Revenue"
```

Step 5: Create Allocations

After you add the required measures, you may want to specify how they will be allocated across the various dimensions in your model. Only do this if you do not use the `ModelEnsureCompleteness` script to generate the default measure allocations.

Alloc_Me.mdl

This script allocates Product Plan summary data to both the Product Line and Year levels of their respective dimensions proportionate to the product Revenue distribution.

```
AllocationAdd Measure "Revenue" Type Default
AllocationAdd Measure "Product Cost" Type Default
AllocationAdd Measure "Product Plan" Type Default
AllocationAdd Drill "By Product Line" Levels "Product
Line"
Measure "Product Plan" Type Allocate
AllocationMeasure "Revenue"
AllocationAdd Dimension "Years" Drill "By Time" Levels
"Year"
Measure "Product Plan" Type Allocate AllocationMeasure
"Revenue"
```

Step 6: Create the Signon

These scripts create Cognos 8 signons and data source signons.

CognosSignon.mdl

This sample script creates a Cognos signon object and disables automatic signon access.

```
SignonAdd "name" UserId "user name" password "password"
AutoLogon False SignonNamespace "namespace name" SignonType
"Cognos"
```

DataSourceSignon.mdl

This sample script creates a data source signon object and disables interactive prompting for a password.

```
SignonAdd "signon name" PromptForPassword True
UserId "user name"
password <password> SignonType "DataSource"
```

Data sources defined in IBM Cognos 8 can have multiple connections, each with multiple signons. You can use MDL to create, delete, and update connection and signon information for ambiguous connections and signons.

You may want to use a script to create or later update the object required to provide automatic signon access.

Step 7: Populate the Model and Create the PowerCube

This is a three-stage process. We recommend that you create separate script files for each process.

Populate.mdl

This script generates categories for the model.

```
Populatemodel
```

Cube.mdl

This script adds the PowerCube object to the model, specifying where the .mdc file is located.

```
CubeAdd "Great Outdoors Sales" MdcFile "c:\Outdoors.mdc"
MeasureInclude 195 Yes
```

Createcube.mdl

This script generates the PowerCube in the form of an .mdc file.

```
CreateFiles
```

Step 8: Create the Model by Combining the Scripts

As a final step, prepare a script that, when run, combines all the previous scripts to create a complete Transformer model and generates an .mdc file based on that model.

NewModel.mdl

This script runs all the previously created scripts in sequence. We recommend that you ensure model completeness before you generate your final .mdl file based on the model.

```
NewModel "ModelScript"
OpenMDL "C:\Que_Col.mdl"
OpenMDL "C:\Dim_Years.mdl"
OpenMDL "C:\Dim_Products.mdl"
OpenMDL "C:\Me_Revenue.mdl"
OpenMDL "C:\Me_Prodcost.mdl"
OpenMDL "C:\Me_Prodplan.mdl"
OpenMDL "C:\Alloc_Me.mdl"
OpenMDL "C:\CognosSignon.mdl"
OpenMDL "C:\DataSourceSignon.mdl"
OpenMDL "C:\Populate.mdl"
OpenMDL "C:\Cube.mdl"
ModelEnsureCompleteness
OpenMDL "C:\Createcube.mdl"
SaveMDL "C:\FinalModel.mdl"
```

Checking the Model Generated by Transformer

When you check the FinalModel.mdl file that results from running your NewModel.mdl script, you will note that the model definition is much longer than it was when you started. This is because

it contains all the default options that Transformer automatically adds to each object definition. In addition,

- the verb `PopulateModel` adds categories, all of which are defined
- the verb `ModelEnsureCompleteness` adds the two default views

Line numbers are included to make the file easier to read. The cogtr.xml setting `ObjectIdOutput=0` is used, hiding the object identifiers by default, so each object is identified by its name alone.

FinalModel.mdl

Below is a line-by-line rendering of the sample mdl file.

No.	MDL
2	DataSourceMake "Products (CSV)" Separator "," SourceType FlatFile_ColNames CharacterSet Ansi DecimalSep "." Thousandsep "," Columns True Timing PopYesCreateDefault Source "c:\prodinfo.csv" EnableMultiProcess False SetCurrent True ServerSource False Speed False Presummarized False
3	ColumnMake "Product Line" DataSource "Products (CSV)" Origin Generated Offset 0 Column "Product Line" Storage Default Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off
4	ColumnMake "Product Type" DataSource "Products (CSV)" Origin Generated Offset 1 Column "Product Type" Storage Default Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off
5	ColumnMake "Product Id" DataSource "Products (CSV)" Origin Generated Offset 2 Column "Product Id" Storage Default Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off
6	ColumnMake "Product Name" DataSource "Products (CSV)" Origin Generated Offset 3 Column "Product Name" Storage Default Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off
7	DataSourceMake "Prod Line Plan" Separator "," SourceType FlatFile_ColNames CharacterSet Ansi DecimalSep "." Thousandsep "," Columns True Timing PopYesCreateDefault Source "c:\ProdPlan.asc" EnableMultiProcess False SetCurrent True ServerSource False Speed False Presummarized False
8	ColumnMake "YEAR" DataSource "Prod Line Plan" Origin Generated Offset 0 Column "Time" Storage Default Format Y DateLevel Year Scale 0 Size 1 Decimals 0 Class Date InputScale 0 TimeArray Off

No.	MDL
9	ColumnMake "PROD_LINE" DataSource "Prod Line Plan" Origin Generated Offset 1 Column "Product Line" Storage Default Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off
10	ColumnMake "FORECAST" DataSource "Prod Line Plan" Origin Generated Offset 2 Column "Planned sales" Storage Default Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off
11	DataSourceMake "MAIN (IQD)" Separator ","SourceType DataSource CharacterSet Ansi DecimalSep ". "Thousandsep "," Columns TrueTiming PopYesCreateDefault Source"c:<install_directory>\bsc_msrs.iqd"SQL 'select T1. "ORDER_DT" as c1 T2."PROD_NO" as c2, T1."REP_NO" as ' ' c3, T1."CUST_NO" as c4, T2."QTY" * T2."PRI' 'CE"as c6, (T2."QTY" * T3. "PROD_COST") as c7, '2."QTY" * T2."PRICE") - (T2."QTY" * T3."PROD_COST"))/ (T2."QTY" * T2." ' 'PRICE")) <= 0.19) THEN (' "'Under 20%')WHEN ((((T2." '"QTY" * T2."PRICE") - (T2."QTY" * T3."PROD_COST"))/ (T2."QTY" * T2."PR' 'ICE")) BETWEEN 0.2 AND 0.65)THEN (' "'20%- 65%') WHEN ((((T2." '"QTY" * T2."PRICE") - (T2."QTY"* T3."PROD_COST")) / (T2."QTY" * T2."PR' 'ICE"))>= 0.66) THEN (' "'Over 65%') ELSE ('ERROR')ENDas c8 from " '"ORDER" T1, ("PRODUCT" T3 left outer join "ORDRDETL" T2 on T2."PROD_NO'" = T3."PROD_NO") order by c1 asc' ",c3 asc,c4 asc,c2asc" Isolation 0 UserEditable FalseSourceSignonList "I70_DBASE_NTV_PP70_SAMPLE" EndListImrName "C:<install_directory>\bsc_msrs.imr" Stamp 996601938 EnableMultiProcess False SetCurrent TrueServerSource False Speed False Presummarized False
12	ColumnMake "Order Dt" DataSource "MAIN (IQD)" Origin Source Offset 0 Column "Time" Storage Int32 Scale 0 Size 4 Decimals 0 Class Date InputScale 0 TimeArray Off
13	ColumnMake "od-Prod No" DataSource "MAIN (IQD)" Origin Source Offset 1 Column "Product Id" Storage Float64 Scale 0 Size 5 Decimals 0 Class Quantity InputScale 0 TimeArray Off
14	ColumnMake "Revenue" DataSource "MAIN (IQD)" Origin Generated Offset 5 Column "Revenue" Storage Default Scale 0 Size 1 Decimals 0 InputScale 0 TimeArray Off

No.	MDL
15	ColumnMake "Cost" DataSource "MAIN (IQD)" Origin Source Offset 6 Column "Cost" Storage Float64 Scale 0 Size 6 Decimals 0 Class Quantity InputScale 0 TimeArray Off
16	DimMake "Years" DimType NonStandard EarliestDate 19000101 LatestDate 99991231 ManualPeriods False DaysInWeek 127 NewCatsLock False ExcludeAutoPartitioning False
17	RootCatMake "Time" Dimension "Years" Inclusion Generate Lastuse 20060802 Date 0 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
18	DrillCatMake "By Time" Dimension "Years" Root "Time" Inclusion Suppress Filtered False Suppressed True PrimaryDrill True YearBegins 20060101 PartialWeek Split ExtraWeek None WeekBegins Sunday
19	LevelMake "Year" Dimension "Years" Drill "By Time" Parent "" Blanks "(blank)" DateFunction Year Generate Need RefreshLabel False RefreshDescription False RefreshShortName False NewCatsLock False CatLabFormat "YYYY" Timerank 10 UniqueCategories True UniqueMove False Associations "Time" AssociationContext "By Time" AssociationType Type_Query AssociationRole Role_OrderBy AssociationReferenced "Time" SortOrder Default SortAs Ascending Associations "Time" AssociationType Type_Query AssociationRole Role_Source AssociationReferenced "Time"
20	LevelMake "Quarter" Dimension "Years" Drill "By Time" Parent "Year" Blanks "(blank)" DateFunction Quarter Generate Need RefreshLabel False RefreshDescription False RefreshShortName False NewCatsLock False CatLabFormat 'YYYY "Q" Q' Timerank 20 UniqueCategories True UniqueMove False Associations "Time" AssociationContext "By Time" AssociationType Type_Query AssociationRole Role_OrderBy AssociationReferenced "Time" SortOrder Default SortAs Ascending Associations "Time" AssociationType Type_Query AssociationRole Role_Source AssociationReferenced "Time"
21	LevelMake "Month" Dimension "Years" Drill "By Time" Parent "Quarter" Blanks "(blank)" DateFunction Month Generate Need RefreshLabel False RefreshDescription False RefreshShortName False NewCatsLock False CatLabFormat "YYYY/MMM" Timerank 30 UniqueCategories True UniqueMove False Associations "Time" AssociationContext "By Time" AssociationType Type_Query AssociationRole Role_OrderBy AssociationReferenced "Time" SortOrder Default SortAs Ascending Associations "Time" AssociationType Type_Query AssociationRole Role_Source AssociationReferenced "Time"

No.	MDL
22	CatMake "20010101-20011231" Dimension "Years" Drill "By Time" Levels "Year" Parent "By Time" OrderBy Drill "By Time" Value "2001" Label "2001" Lastuse 20060802 SourceValue "2001" Date 20010101 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
23	CatMake "20010101-20010331" Dimension "Years" Drill "By Time" Levels "Quarter" Parent "20010101-20011231" OrderBy Drill "By Time" Value "20010101" Label "2001 Q 1" Lastuse 20060802 SourceValue "20010101" Date 20010101 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
24	CatMake "20010101-20010131" Dimension "Years" Drill "By Time" Levels "Month" Parent "20010101-20010331" OrderBy Drill "By Time" Value "200101" Label "2001/Jan" Lastuse 20060802 SourceValue "200101" Date 20010101 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
25	CatMake "20010201-20010229" Dimension "Years" Drill "By Time" Levels "Month" Parent "20010101-20010331" OrderBy Drill "By Time" Value "200102" Label "2001/Feb" Lastuse 20060802 SourceValue "200102" Date 20010201 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
26	CatMake "20010301-20010331" Dimension "Years" Drill "By Time" Levels "Month" Parent "20010101-20010331" OrderBy Drill "By Time" Value "200103" Label "2001/Mar" Lastuse 20060802 SourceValue "200103" Date 20010301 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
27	CatMake "20010401-20010630" Dimension "Years" Drill "By Time" Levels "Quarter" Parent "20010101-20011231" OrderBy Drill "By Time" Value "20010401" Label "2001 Q 2" Lastuse 20060802 SourceValue "20010401" Date 20010401 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
28	CatMake "20010401-20010430" Dimension "Years" Drill "By Time" Levels "Month" Parent "20010401-20010630" OrderBy Drill "By Time" Value "200104" Label "2001/Apr" Lastuse 20060802 SourceValue "200104" Date 20010401 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False

No.	MDL
29	CatMake "20010501-20010531" Dimension "Years" Drill "By Time" Levels "Month" Parent "20010401-20010630" OrderBy Drill "By Time" Value "200105" Label "2001/May" Lastuse 20060802 SourceValue "200105" Date 20010501 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
30	CatMake "20010601-20010630" Dimension "Years" Drill "By Time" Levels "Month" Parent "20010401-20010630" OrderBy Drill "By Time" Value "200106" Label "2001/Jun" Lastuse 20060802 SourceValue "200106" Date 20010601 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
31	CatMake "20010701-20010930" Dimension "Years" Drill "By Time" Levels "Quarter" Parent "20010101-20011231" OrderBy Drill "By Time" Value "20010701" Label "2001 Q 3" Lastuse 20060802 SourceValue "20010701" Date 20010701 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
32	CatMake "20010701-20010731" Dimension "Years" Drill "By Time" Levels "Month" Parent "20010701-20010930" OrderBy Drill "By Time" Value "200107" Label "2001/Jul" Lastuse 20060802 SourceValue "200107" Date 20010701 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
33	CatMake "20010801-20010831" Dimension "Years" Drill "By Time" Levels "Month" Parent "20010701-20010930" OrderBy Drill "By Time" Value "200108" Label "2001/Aug" Lastuse 20060802 SourceValue "200108" Date 20010801 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
34	CatMake "20010901-20010930" Dimension "Years" Drill "By Time" Levels "Month" Parent "20010701-20010930" OrderBy Drill "By Time" Value "200109" Label "2001/Sep" Lastuse 20060802 SourceValue "200109" Date 20010901 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
35	CatMake "20011001-20011231" Dimension "Years" Drill "By Time" Levels "Quarter" Parent "20010101-20011231" OrderBy Drill "By Time" Value "20011001" Label "2001 Q 4" Lastuse 20060802 SourceValue "20011001" Date 20011001 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False

No.	MDL
36	CatMake "20011001-20011031" Dimension "Years" Drill "By Time" Levels "Month" Parent "20011001-20011231" OrderBy Drill "By Time" Value "200110" Label "2001/Oct" Lastuse 20060802 SourceValue "200110" Date 20011001 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
37	CatMake "20011101-20011130" Dimension "Years" Drill "By Time" Levels "Month" Parent "20011001-20011231" OrderBy Drill "By Time" Value "200111" Label "2001/Nov" Lastuse 20060802 SourceValue "200111" Date 20011101 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
38	CatMake "20011201-20011231" Dimension "Years" Drill "By Time" Levels "Month" Parent "20011001-20011231" OrderBy Drill "By Time" Value "200112" Label "2001/Dec" Lastuse 20060802 SourceValue "200112" Date 20011201 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
39	CatMake "20020101-20021231" Dimension "Years" Drill "By Time" Levels "Year" Parent "By Time" OrderBy Drill "By Time" Value "2002" Label "2002" Lastuse 20060802 SourceValue "2002" Date 20020101 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
40	CatMake "20020101-20020331" Dimension "Years" Drill "By Time" Levels "Quarter" Parent "20020101-20021231" OrderBy Drill "By Time" Value "20020101" Label "2002 Q 1" Lastuse 20060802 SourceValue "20020101" Date 20020101 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
41	CatMake "20020101-20020131" Dimension "Years" Drill "By Time" Levels "Month" Parent "20020101-20020331" OrderBy Drill "By Time" Value "200201" Label "2002/Jan" Lastuse 20060802 SourceValue "200201" Date 20020101 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
42	CatMake "20020201-20020228" Dimension "Years" Drill "By Time" Levels "Month" Parent "20020101-20020331" OrderBy Drill "By Time" Value "200202" Label "2002/Feb" Lastuse 20060802 SourceValue "200202" Date 20020201 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False

No.	MDL
43	CatMake "20020301-20020331" Dimension "Years" Drill "By Time" Levels "Month" Parent "20020101-20020331" OrderBy Drill "By Time" Value "200203" Label "2002/Mar" Lastuse 20060802 SourceValue "200203" Date 20020301 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
44	CatMake "20020401-20020630" Dimension "Years" Drill "By Time" Levels "Quarter" Parent "20020101-20021231" OrderBy Drill "By Time" Value "20020401" Label "2002 Q 2" Lastuse 20060802 SourceValue "20020401" Date 20020401 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
45	CatMake "20020401-20020430" Dimension "Years" Drill "By Time" Levels "Month" Parent "20020401-20020630" OrderBy Drill "By Time" Value "200204" Label "2002/Apr" Lastuse 20060802 SourceValue "200204" Date 20020401 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
46	CatMake "20020501-20020531" Dimension "Years" Drill "By Time" Levels "Month" Parent "20020401-20020630" OrderBy Drill "By Time" Value "200205" Label "2002/May" Lastuse 20060802 SourceValue "200205" Date 20020501 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
47	CatMake "20020601-20020630" Dimension "Years" Drill "By Time" Levels "Month" Parent "20020401-20020630" OrderBy Drill "By Time" Value "200206" Label "2002/Jun" Lastuse 20060802 SourceValue "200206" Date 20020601 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
48	CatMake "20020701-20020930" Dimension "Years" Drill "By Time" Levels "Quarter" Parent "20020101-20021231" OrderBy Drill "By Time" Value "20020701" Label "2002 Q 3" Lastuse 20060802 SourceValue "20020701" Date 20020701 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
49	CatMake "20020701-20020731" Dimension "Years" Drill "By Time" Levels "Month" Parent "20020701-20020930" OrderBy Drill "By Time" Value "200207" Label "2002/Jul" Lastuse 20060802 SourceValue "200207" Date 20020701 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False

No.	MDL
50	CatMake "20020801-20020831" Dimension "Years" Drill "By Time" Levels "Month" Parent "20020701-20020930" OrderBy Drill "By Time" Value "200208" Label "2002/Aug" Lastuse 20060802 SourceValue "200208" Date 20020801 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
51	CatMake "20020901-20020930" Dimension "Years" Drill "By Time" Levels "Month" Parent "20020701-20020930" OrderBy Drill "By Time" Value "200209" Label "2002/Sep" Lastuse 20060802 SourceValue "200209" Date 20020901 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
52	CatMake "20021001-20021231" Dimension "Years" Drill "By Time" Levels "Quarter" Parent "20020101-20021231" OrderBy Drill "By Time" Value "20021001" Label "2002 Q 4" Lastuse 20060802 SourceValue "20021001" Date 20021001 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
53	CatMake "20021001-20021031" Dimension "Years" Drill "By Time" Levels "Month" Parent "20021001-20021231" OrderBy Drill "By Time" Value "200210" Label "2002/Oct" Lastuse 20060802 SourceValue "200210" Date 20021001 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
54	CatMake "20021101-20021130" Dimension "Years" Drill "By Time" Levels "Month" Parent "20021001-20021231" OrderBy Drill "By Time" Value "200211" Label "2002/Nov" Lastuse 20060802 SourceValue "200211" Date 20021101 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
55	CatMake "20021201-20021231" Dimension "Years" Drill "By Time" Levels "Month" Parent "20021001-20021231" OrderBy Drill "By Time" Value "200212" Label "2002/Dec" Lastuse 20060802 SourceValue "200212" Date 20021201 Current Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
56	ViewMake "All Categories" Dimension "Years" Type All
57	ViewMake "Omit Dimension" Dimension "Years" Type Omit
58	DimMake "Products" DimType Regular NewCatsLock False ExcludeAutoPartitioning False RootCatMake "Product Line" Dimension "Products" Inclusion Generate Lastuse 20060802 Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False

No.	MDL
59	DrillCatMake "By Product Line" Dimension "Products" Root "Product Line" Inclusion Suppress Filtered False Suppressed True PrimaryDrill True
60	LevelMake "Product Line" Dimension "Products" Drill "By Product Line" Parent "" Blanks "(blank)" DateFunction None Generate Need RefreshLabel False RefreshDescription False RefreshShortName False NewCatsLock False Timerank 0 UniqueCategories False UniqueMove False Associations "Product Line" AssociationType Type_Query AssociationRole Role_Source AssociationReferenced "Product Line"
61	LevelMake "Product Type" Dimension "Products" Drill "By Product Line" Parent "Product Line" Blanks "(blank)" DateFunction None Generate Need RefreshLabel False RefreshDescription False RefreshShortName False NewCatsLock False Timerank 0 UniqueCategories False UniqueMove False Associations "Product Type" AssociationType Type_Query AssociationRole Role_Source AssociationReferenced "Product Type"
62	LevelMake "Product Id" Dimension "Products" Drill "By Product Line" Parent "Product Type" Blanks "(blank)" DateFunction None Generate Need RefreshLabel False RefreshDescription False RefreshShortName False NewCatsLock False Timerank 0 UniqueCategories True UniqueMove False Associations "Product Id" AssociationType Type_Query AssociationRole Role_Source AssociationReferenced "Product Id" Associations "Product Name" AssociationType Type_Query AssociationRole Role_Label AssociationReferenced "Product Name"
63	CatMake "Environmental Line" Dimension "Products" Drill "By Product Line" Levels "Product Line" Parent "By Product Line" Lastuse 20060802 SourceValue "Environmental Line" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
64	CatMake "Alert Devices" Dimension "Products" Drill "By Product Line" Levels "Product Type" Parent "Environmental Line" Lastuse 20060802 SourceValue "Alert Devices" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
65	CatMake "60100" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Alert Devices" Label "Pocket U.V. Alerter" Lastuse 20060802 SourceValue "60100" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False

No.	MDL
66	CatMake "60101" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Alert Devices" Label "Microwave Detector" Lastuse 20060802 SourceValue "60101" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
67	CatMake "60102" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Alert Devices" Label "Pocket Radon Alerter" Lastuse 20060802 SourceValue "60102" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
68	CatMake "Bio-Friendly Soaps" Dimension "Products" Drill "By Product Line" Levels "Product Type" Parent "Environmental Line" Lastuse 20060802 SourceValue "Bio-Friendly Soaps" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
69	CatMake "60300" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Bio-Friendly Soaps" Label "RiverKind Shampoo" Lastuse 20060802 SourceValue "60300" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
70	CatMake "60301" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Bio-Friendly Soaps" Label "RiverKind Soap" Lastuse 20060802 SourceValue "60301" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
71	CatMake "60302" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Bio-Friendly Soaps" Label "RiverKind Detergent" Lastuse 20060802 SourceValue "60302" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
72	CatMake "Recycled Products" Dimension "Products" Drill "By Product Line" Levels "Product Type" Parent "Environmental Line" Lastuse 20060802 SourceValue "Recycled Products" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
73	CatMake "60201" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Recycled Products" Label "Enviro-Kit" Lastuse 20060802 SourceValue "60201" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
74	CatMake "60200" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Recycled Products" Label "EnviroSak" Lastuse 20060802 SourceValue "60200" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False

No.	MDL
75	CatMake "60202" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Recycled Products" Label "Enviro-T" Lastuse 20060802 SourceValue "60202" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
76	CatMake "Sunblock" Dimension "Products" Drill "By Product Line" Levels "Product Type" Parent "Environmental Line" Lastuse 20060802 SourceValue "Sunblock" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
77	CatMake "60400" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Sunblock" Label "Sun Shelter-8" Lastuse 20060802 SourceValue "60400" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
78	CatMake "60401" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Sunblock" Label "Sun Shelter-15" Lastuse 20060802 SourceValue "60401" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
79	CatMake "60402" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Sunblock" Label "Sun Shelter-30" Lastuse 20060802 SourceValue "60402" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
80	CatMake "Water Purifiers" Dimension "Products" Drill "By Product Line" Levels "Product Type" Parent "Environmental Line" Lastuse 20060802 SourceValue "Water Purifiers" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
81	CatMake "60500" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Water Purifiers" Label "Pro-Lite Water Filter" Lastuse 20060802 SourceValue "60500" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
82	CatMake "60501" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Water Purifiers" Label "Pocket Water Filter" Lastuse 20060802 SourceValue "60501" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
83	CatMake "GO Sport Line" Dimension "Products" Drill "By Product Line" Levels "Product Line" Parent "By Product Line" Lastuse 20060802 SourceValue "GO Sport Line" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False

No.	MDL
84	CatMake "Carry-Bags" Dimension "Products" Drill "By Product Line" Levels "Product Type" Parent "GO Sport Line" Lastuse 20060802 SourceValue "Carry-Bags" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
85	CatMake "50100" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Carry-Bags" Label "GO Sport Bag" Lastuse 20060802 SourceValue "50100" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
86	CatMake "50101" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Carry-Bags" Label "GO Ski Gear Bag" Lastuse 20060802 SourceValue "50101" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
87	CatMake "50102" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Carry-Bags" Label "GO Duffel Bag" Lastuse 20060802 SourceValue "50102" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
88	CatMake "Sport Wear" Dimension "Products" Drill "By Product Line" Levels "Product Type" Parent "GO Sport Line" Lastuse 20060802 SourceValue "Sport Wear" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
89	CatMake "50201" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Sport Wear" Label "GO Headband" Lastuse 20060802 SourceValue "50201" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
90	CatMake "50201" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Sport Wear" Label "GO Headband" Lastuse 20060802 SourceValue "50201" Filtered False S uppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
91	CatMake "50202" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Sport Wear" Label "GO Wristband" Lastuse 20030825 SourceValue "50202" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
92	CatMake "50203" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Sport Wear" Label "GO Water Bottle" Lastuse 20030825 SourceValue "50203" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False

No.	MDL
93	CatMake "Outdoor Products" Dimension "Products" Drill "By Product Line" Levels "Product Line" Parent "By Product Line" Lastuse 20060802 SourceValue "Outdoor Products" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
94	CatMake "Back Packs" Dimension "Products" Drill "By Product Line" Levels "Product Type" Parent "Outdoor Products" Lastuse 20060802 SourceValue "Back Packs" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
95	CatMake "40300" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Back Packs" Label "Day Tripper" Lastuse 20060802 SourceValue "40300" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
96	CatMake "40301" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Back Packs" Label "Pack n' Hike" Lastuse 20060802 SourceValue "40301" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
97	CatMake "40302" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Back Packs" Label "GO Small Waist Pack" Lastuse 20060802 SourceValue "40302" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
98	CatMake "40303" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Back Packs" Label "GO Large Waist Pack" Lastuse 20060802 SourceValue "40303" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
99	CatMake "Cooking Equipment" Dimension "Products" Drill "By Product Line" Levels "Product Type" Parent "Outdoor Products" Lastuse 20060802 SourceValue "Cooking Equipment" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
100	CatMake "40400" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Cooking Equipment" Label "Dover-1" Lastuse 20060802 SourceValue "40400" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
101	CatMake "40401" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Cooking Equipment" Label "Dover-2" Lastuse 20060802 SourceValue "40401" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False

No.	MDL
102	CatMake "40402" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Cooking Equipment" Label "GO Cookset" Lastuse 20060802 SourceValue "40402" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
103	CatMake "40403" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Cooking Equipment" Label "GO Camp Kettle" Lastuse 20060802 SourceValue "40403" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
104	CatMake "Sleeping Bags" Dimension "Products" Drill "By Product Line" Levels "Product Type" Parent "Outdoor Products" Lastuse 20060802 SourceValue "Sleeping Bags" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
105	CatMake "40200" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Sleeping Bags" Label "MoonBeam" Lastuse 20060802 SourceValue "40200" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
106	CatMake "40201" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Sleeping Bags" Label "MoonGlow" Lastuse 20060802 SourceValue "40201" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
107	CatMake "40202" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Sleeping Bags" Label "MoonLite" Lastuse 20060802 SourceValue "40202" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
108	CatMake "Tents" Dimension "Products" Drill "By Product Line" Levels "Product Type" Parent "Outdoor Products" Lastuse 20060802 SourceValue "Tents" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
109	CatMake "40100" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Tents" Label" Star Lite" Lastuse 20060802 SourceValue "40100" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
110	CatMake "40101" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Tents" Label "Star Gazer-2" Lastuse 20060802 SourceValue "40101" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False

No.	MDL
111	CatMake "40102" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Tents" Label "Star Gazer-3" Lastuse 20060802 SourceValue "40102" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
112	CatMake "40103" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Tents" Label "StarDome" Lastuse 20060802 SourceValue "40103" Current Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
113	CatMake "40104" Dimension "Products" Drill "By Product Line" Levels "Product Id" Parent "Tents" Label "Micro Lite" Lastuse 20060802 SourceValue "40104" Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated False Blanks False
114	ViewMake "All Categories" Dimension "Products" Type All
115	ViewMake "Omit Dimension" Dimension "Products" Type Omit
116	MeasureMake "Revenue" Storage Default OutPutScale 0 Decimals 0 ReverseSign False IsCurrency False DrillThrough False EndList Associations "Revenue" AssociationType Type_Query AssociationRole Role_Source AssociationReferenced "Revenue"
117	MeasureMake "Product Cost" Storage Default OutPutScale 0 Decimals 0 ReverseSign False IsCurrency False DrillThrough False EndList Associations "Cost" AssociationType Type_Query AssociationRole Role_Source AssociationReferenced "Cost"
118	MeasureMake "Product Plan" Storage Default OutPutScale 0 Decimals 0 ReverseSign False IsCurrency False DrillThrough False EndList Associations "Planned Sales" AssociationType Type_Query AssociationRole Role_Source AssociationReferenced "Planned sales"

No.	MDL
119	SignonMake "I70_DBASE_NTV_PP70_SAMPLE"FullDb" 7C3155E28283DDFEF647A376504802295E3F3A5FACBCD1616F8770A97C 530151BD5644""09010B9D440F8CC047E5E62D7968B87633C7BC6DA56D9539D5 FB47D8A1A5E4EE9790A0""A00CDC111C1CF518BA09F9DAD8971B53E3D1DCA8A2 2962F223FA52F4CCE534946A57A5""23FF8FA4643644A41E7F18519B37C7E8C8 81B2A80ADAFAB3A3053D4D728B8BAA51F49B0""C4CCB26314FF28DB56ED4BF73 57503BAFF0EB65F0713691B4FC64ED26372563E2042978""2A618F81AF735920 8330F265CAD582A19D9200E78660EC690A4EF3186AFEE7A3E7429A8""702C51C 693EF7EA7F3D673E4B8C569D51E254781B0C6A3CAAA6ED4CD5F94AC450E79526 ""1C53078EFF0ECC"DbType "DB" DatabaseSubType BDEPromptForPassword False Manual False SignonMake "I50_DBASE_NTV_ PP60_SAMPLE" PromptForPassword FalseManual False
120	CubeMake "Great Outdoors Sales" MdcFile "c:\Outdoors.mdc" Status OK CubeCreation On Optimize Categories Compress False DatabaseInfo "Local;" IncrementalUpdate False ServerCube False CubeStamp 996769127 CubeCycle 2 BlockParentTotals False Caching False UseAlternateFileName False DrillThrough False EndList DimensionView "Years" "All Categories" DimensionView "Products" "All Categories" MeasureInclude "Revenue" Yes MeasureInclude "Product Cost" Yes MeasureInclude "Product Plan" Yes
121	AllocationAdd Measure "Revenue" Type Default
122	AllocationAdd Measure "Product Cost" Type Default
123	AllocationAdd Measure "Product Plan" Type Default
124	AllocationAdd Dimension "Products" Drill "By Product Line" Levels "Product Line" Measure "Product Plan" Type Allocate AllocationMeasure "Revenue"
125	AllocationAdd Dimension "Years" Drill "By Time" Levels "Year" Measure "Product Plan" Type Allocate AllocationMeasure "Revenue"

Example - MDL Model Using an IBM Cognos 8 Data Source in Structured MDL

This is the order in which you would create a typical model using structured MDL:

- Create the data sources in Transformer.
- Create the dimensions and key performance measures.

- Create the standard dimension views.
- Create dimension views and custom views.
- Add the namespace and security objects.
- Populate the model and create the PowerCube.

The following example shows a model, created using an IBM Cognos 8 data source, in structured MDL format.

Structured MDL	Description
CognosSource 103 **"GO Sales and Retailers" SourceType** Package **SourcePath** "/content/package[@name='GO Sales and Retailers']" **PackageTimeStamp** "/content/package[@name='GO Sales and Retailers']/model[@name='model']"	**CognosSource** defines an IBM Cognos 8 package or report as the data source in the model. **"Go Sales and Retailers"** is the name of the IBM Cognos 8 package. All packages and queries must have unique names. **SourceType** defines the type of IBM Cognos 8 data source. The type can be Package, Report, or CognosSourceQuery. In this example, SourceType defines a package data source type. **SourcePath** is the path to the IBM Cognos 8 package stored in Content Manager. **PackageTimeStamp** is the version of the package last used in the model.
CognosSource 11543 "QuantityReport" **SourceType** Report SourcePath "/content/package[@name='GO Sales and Retailers']/report[@name='QuantityReport']" PackageTimeStamp "/content/package[@name='GO Sales and Retailers']/report[@name='QuantityReport']"	**CognosSource** defines a second IBM Cognos 8 package or report data source in the model. In this example, the **SourceType** defines a report data source type.
DataSource 105 "GO Sales and Retailers~1" Separator "," SourceType **CognosSourceQuery** CharacterSet Default DecimalSep "." Thousandsep "," Columns True Timing PopYesCreateDefault **PackageReportSource 103 "GO Sales and Retailers"** AutoSummary True SetCurrent True ServerSource False Speed False Presummarized False	**CognosSourceQuery** defines a query defined against a package or report SourceType. **PackageReportSource 103 "GO Sales and Retailers"** is the reference to the package on which this query is defined.

Structured MDL	Description
OrgName 107 **"[gosales_goretailers].[Orders].[Order number]"** Origin Source Offset 0 Column "Order number" Storage Float64 Scale 0 Size 4 Decimals 0 InputScale 0 TimeArray Off Rollup CountAll OrgName 109 "[gosales_goretailers]. [Orders].[Retailer name]" Origin Source Offset 1 Column "Retailer name" Storage Text Scale 0 Size 102 Decimals 0 Class Description InputScale 0 TimeArray Off OrgName 111 "[gosales_goretailers].[Orders]. [Order date]" Origin Source Offset 2 Column "Order date" Storage Int32 Scale 0 Size 12 Decimals 0 Class Date InputScale 0 TimeArray Off OrgName 113 "[gosales_goretailers].[Orders]. [Order method]" Origin Source Offset 3 Column "Order method" Storage Text Scale 0 Size 102 Decimals 0 Class Description InputScale 0 TimeArray Off OrgName 115 "[gosales_goretailers]. [Orders].[Order method code]" Origin Source Offset 4 Column "Order method code" Storage Float64 Scale 0 Size 4 Decimals 0 InputScale 0 TimeArray Off Rollup CountAll OrgName 117 "[gosales_goretailers].[Orders].[Product name]"Origin Source Offset 5 Column "Product name" Storage Text Scale 0 Size 102 Decimals 0 Class Description InputScale 0 TimeArray Off OrgName 119 "[gosales_goretailers].[Orders]. [Quantity]" Origin Source Offset 6 Column "Quantity" Storage Float64 Scale 0 Size 2 Decimals 0 Class Quantity InputScale 0 TimeArray Off Rollup Sum	**"[gosales_goretailers].[Orders].[Order number]"** is the reference within the package or report to the query item or measure on which this column is defined. Following are more references to query items or measures on which the columns are defined.

Structured MDL	Description
Filter 11553 "Americas" **FilterRef "[gosales_goretailers].[Americas]"**	**Filter 115533 "Americas"** selects a filter from the data source and imports it into the query. **FilterRef "[gosales_goretailers].[Americas]"** is the reference within the package or report to the filter on which this column is defined.
DataSource 11545 "QuantityReport~1" Separator "," SourceType CognosSourceQuery CharacterSet Default DecimalSep "." Thousandsep "," Columns True Timing PopYesCreateDefault **PackageReportSource 11543 "QuantityReport"** AutoSummary False SetCurrent True ServerSource False Speed False Presummarized False OrgName 11547 "[Report].[Query1.0].[Product number]" Origin Source Offset 0 Column "Product number" Storage Float64 Scale 0 Size 1 Decimals 0 Class Quantity InputScale 0 TimeArray Off OrgName 11549 "[Report].[Query1.0].[Production cost]" Origin Source Offset 1 Column "Production cost" Storage Float64 Scale 2 Size 1 Decimals 2 Class Quantity InputScale 0 TimeArray Off Dimension 149 "Order date" DimType Date EarliestDate 19010101 LatestDate 21001231 ManualPeriods False DaysInWeek 127 NewCatsLock False ExcludeAutoPartitioning False DimDefaultCategory 0	**PackageReportSource 11543 "QuantityReport"** is a second query, defined against a second package. Following are more references to query items or measures on which the columns are defined.

Structured MDL	Description
Categories Root 153 "Order date" Inclusion Generate Lastuse 20070910 Date 0 Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False Drill 155 "By Order date" Inclusion Suppress Filtered False Suppressed True PrimaryDrill True HideValue False YearBegins 20070101 PartialWeek Split ExtraWeek None WeekBegins Sunday Levels 161 "Year" Blanks "(blank)" Inclusion Generate DateFunction Year Generate Need RefreshLabel False RefreshDescription False RefreshShortName False NewCatsLock False CatLabFormat "YYYY" Timerank 10 **UniqueCategories True** UniqueMove False Associations 163 "Order date" AssociationType Type_Query AssociationRole Role_Source AssociationReferenced "Order date" Associations 165 "Order date" AssociationContext 155 AssociationType Type_Query AssociationRole Role_OrderBy AssociationReferenced "Order date" SortOrder Int16 SortAs Ascending	**UniqueCategories True** indicates that the Product Id level is unique. This is necessary because the level is associated with columns from multiple queries.

Structured MDL	Description

```
Levels 167 "Quarter" Blanks "( blank)"

Inclusion Generate DateFunction Quarter

Generate All RefreshLabel False

RefreshDescription False Refresh
ShortName False NewCatsLock False CatLabFormat 'YYYY
 "Q"
Q'

Timerank 20 UniqueCategories True

UniqueMove False

Associations 169 "Order date"

AssociationType Type_Query

AssociationRole Role_Source

AssociationReferenced "Order date"

Associations 171 "Order date"

AssociationContext 155

AssociationType Type_Query

AssociationRole Role_OrderBy

AssociationReferenced "Order date"
SortOrder Int16 SortAs Ascending

Category 233 "20040101-20041231"

Parent 155 Levels 161 OrderBy Drill 155 Value "2004"

Label "2004" Lastuse 20070910 SourceValue "2004" Date
 20040101 Filtered
False
Suppressed False Sign False HideValue False
IsKeyOrphanage False
IsTruncated False
Blanks False Category 235 "20040101-20040331" Parent
 233
Levels 167 OrderBy Drill 155
Value "20040101" Label "2004 Q 1" Lastuse 20070910
SourceValue "20040101"
Date 20040101 Filtered False Suppressed False
Sign False
HideValue False
IsKeyOrphanage False IsTruncated False
Blanks False

Category 243 "20040401-20040630"
Parent 233 Levels 167 OrderBy Drill 155 Value
"20040401" Label "2004
Q 2" Lastuse 20070910 SourceValue "20040401" Date
20040401 Filtered
False
Suppressed False Sign False HideValue False
IsKeyOrphanage False
IsTruncated False Blanks False
```

Structured MDL	Description

```
Category 251 "20040701-20040930"
Parent 233 Levels 167 OrderBy Drill 155 Value
"20040701" Label "2004
Q 3" Lastuse 20070910 SourceValue "20040701"
Date 20040701 Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False
IsTruncated False
Blanks False

Category 259 "20041001-20041231"
Parent 233 Levels 167 OrderBy Drill 155 Value
"20041001" Label "2004
Q 4"
Lastuse 20070910 preserve">SourceValue "20041001"
Date 20041001
Filtered False Suppressed False Sign False
HideValue False
IsKeyOrphanage
False IsTruncated False Blanks False

Category 581 "20050101-20051231"
Parent 155 Levels 161 OrderBy Drill 155 Value "2005"
 Label "2005"
Lastuse 20070910 SourceValue "2005" Date 20050101
Filtered False
Suppressed False Sign False HideValue False
IsKeyOrphanage False
IsTruncated False Blanks False Category 583
"20050101-20050331"
Parent 581 Levels 167 OrderBy Drill 155 Value
"20050101" Label "2005
Q 1"
Lastuse 20070910 SourceValue "20050101" Date 20050101
 Filtered False
Suppressed False Sign False HideValue False
IsKeyOrphanage False
IsTruncated False Blanks False

Category 591 "20050401-20050630"
Parent 581 Levels 167 OrderBy Drill 155 Value
"20050401" Label "2005
Q 2"
Lastuse 20070910 SourceValue "20050401" Date 20050401
 Filtered False
Suppressed False Sign False HideValue False
IsKeyOrphanage False
IsTruncated False Blanks False

Category 599 "20050701-20050930"
Parent 581 Levels 167 OrderBy Drill 155 Value
"20050701" Label "2005
Q 3"
Lastuse 20070910 SourceValue "20050701" Date 20050701
 Filtered False
Suppressed False Sign False HideValue False
IsKeyOrphanage False
IsTruncated False Blanks False
```

Structured MDL	Description

```
Category 607 "20051001-20051231"
Parent 581 Levels 167 OrderBy Drill 155 Value
"20051001" Label "2005
Q 4"
Lastuse 20070910 SourceValue "20051001" Date 20051001
 Filtered False
Suppressed
False Sign False HideValue False IsKeyOrphanage False

IsTruncated False Blanks False


Category 683 "20060101-20061231"
Parent 155 Levels 161 OrderBy Drill 155 Value "2006"
 Label "2006"
Lastuse 20070910 SourceValue "2006" Date 20060101
Filtered
False

Suppressed False Sign False
HideValue False IsKeyOrphanage False

IsTruncated False Blanks False
Category 685 "20060101-20060331" Parent 683

Levels 167 OrderBy Drill 155
Value "20060101"

Label "2006 Q 1"
Lastuse 20070910

SourceValue "20060101"
Date 20060101 Filtered False Suppressed False

Sign False HideValue False
IsKeyOrphanage False IsTruncatedFalse Blanks False


Category 693 "20060401-20060630"

Parent 683 Levels 167 OrderBy Drill 155

Value "20060401" Label "2006 Q 2"

Lastuse 20070910 SourceValue "20060401"

Date 20060401

Filtered False Suppressed False Sign False
HideValue False IsKeyOrphanage False IsTruncated False
Blanks False


Category 701 "20060701-20060930"
Parent 683

Levels 167
OrderBy Drill 155 Value "20060701" Label "2006 Q 3"
Lastuse 20070910
SourceValue "20060701"
Date 20060701 Filtered False Suppressed False Sign
False HideValue
False
IsKeyOrphanage False IsTruncated False Blanks False
```

Structured MDL	Description

```
Category 709 "20061001-20061231"
Parent 683
Levels 167 OrderBy Drill 155 Value "20061001"

Label "2006 Q 4" Lastuse 20070910
SourceValue "20061001" Date 20061001

Filtered False Suppressed False
Sign False HideValue False

IsKeyOrphanage False IsTruncated False Blanks False


MapDrills MapDrill 155


ViewName 157 "All Categories" Type
All ViewCustomView 0


ViewName 159 "Omit Dimension" Type
Omit ViewCustomView 0


Dimension 195 "Order method"
DimType Regular NewCatsLock False
ExcludeAutoPartitioning False
DimDefaultCategory 0


Categories Root 197 "Order method"
Inclusion Generate Lastuse 20070910 Filtered False

Suppressed False Sign False
HideValue False IsKeyOrphanage False IsTruncated False

Blanks False Drill 199 "By Order method"

Inclusion Suppress Filtered False

Suppressed True
PrimaryDrill True HideValue False

 Levels 205 "Order method"
Blanks "( Blank )" DateFunction None Generate None

RefreshLabel False
RefreshDescription False RefreshShortName False
NewCatsLock False

Timerank 0 UniqueCategories False
UniqueMove False Associations 207 "Order method code"

AssociationType Type_Query AssociationRole Role_Source


AssociationReferenced "Order method code"


Associations 209 "Order method"
AssociationType Type_Query AssociationRole Role_Label


AssociationReferenced "Order method"


Category 267 "7" Parent 199 Levels
205

Label "Sales visit" Lastuse 20070910

SourceValue "7"
Filtered False Suppressed False Sign False HideValue
 False
IsKeyOrphanage False IsTruncated False Blanks False
```

Structured MDL	Description

```
Category 285 "4" Parent 199
Levels 205 Label "E-mail" Lastuse 20070910 SourceValue
 "4"
Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False
IsTruncated False Blanks False

Category 309 "5" Parent 199 Levels
205
Label "Web" Lastuse 20070910 SourceValue "5" Filtered
 False
Suppressed False Sign False HideValue False
IsKeyOrphanage False
IsTruncated False Blanks False

Category 319 "2" Parent 199 Levels
205
Label "Telephone" Lastuse 20070910 SourceValue "2"
Filtered False
Suppressed False Sign False
HideValue False IsKeyOrphanage False
IsTruncated False Blanks False

Category 357 "1" Parent 199 Levels
205
Label "Fax" Lastuse 20070910 SourceValue "1" Filtered
 False
Suppressed False Sign False HideValue False
IsKeyOrphanage False
IsTruncated False Blanks False

Category 437 "3" Parent 199 Levels
205
Label "Mail" Lastuse 20070910 SourceValue "3" Filtered
 False
Suppressed False Sign False HideValue False
IsKeyOrphanage False
IsTruncated False Blanks False

Category 475 "8" Parent 199 Levels
205 Label "Special"
Lastuse 20070910 SourceValue "8"
Current Filtered False
Suppressed False Sign False HideValue False
IsKeyOrphanage False
IsTruncated False Blanks False

MapDrills MapDrill 199
```

Structured MDL	Description
ViewName 201 "All Categories" Type All ViewCustomView 0 **ViewName 203 "Omit Dimension" Type Omit ViewCustomView 0**	These are the two standard views created for all dimensions. If you do not manually create the two default dimension views required for each dimension (**All Categories** and **Omit Dimension**), you must use the `ModelEnsureCompleteness` script to have Transformer create them automatically.
ViewName 11555 "Authors~User View" ViewCustomView 11529 Apex 197 Filter 319	**ViewName 11555 "Authors~User View"** creates a customized dimension view. **ViewCustomView 11529** is a reference to the view that this custom view uses in the dimension. **Apex 197 Filter 319** is a list of the categories that have customization options set, such as apexing or cloaking.
ViewName 11557 "Sub Authors~User View" ViewCustomView 11537 Apex 197 Filter 319	
Measure 225 "Quantity" Rollup Sum IgnoreMissingValue False Storage Float64 OutPutScale 0 Decimals 0 ReverseSign False IsCurrency False IsFolder False DrillThrough False EndList Associations 227 "Quantity" AssociationType Type_Query AssociationRole Role_Source AssociationReferenced "Quantity"	

Structured MDL	Description
CustomView 11529 **"Authors"** **DimensionView 149 "All Categories"** **DimensionView 195 "Authors~User View"** **MeasureInclude 225 Yes**	**CustomView** indicates that a custom view has been defined. **"Authors"** is the name of the custom view. Custom view names must be unique. **DimensionView 149 "All Categories"** is a reference to a dimension (**149**) and the view that this custom view uses in that dimension, including all categories in the dimension. The second dimension, **DimensionView 195 "Authors~User View"**, is the customized dimension view used for this custom view. **MeasureInclude 225 Yes** indicates that measure **225** is included in the custom view. **MeasureInclude** values are **Yes** or **No**.
CustomView 11537 "Sub Authors" DimensionView 149 "All Categories" DimensionView 195 "Sub Authors~User View"	This entry creates a second custom view.
MeasureInclude 225 No	**MeasureInclude 225 No** indicates that measure **225** is excluded from this custom view.
CustomViewChildList **11529 StartList 11537 EndList** CustomViewChildList 11537 StartList EndList	**CustomViewChildList 11529 StartList 11537 EndList** defines the custom views that are descendents of the custom view "Authors". References to custom views may be unique identifiers or names.
SecurityNameSpace 11533 **"Cognos" SecurityNameSpaceCAMID 'CAMID(":")'**	**SecurityNameSpace...** defines the namespace from which security objects are retrieved. In this example, **"Cognos"** is the name of the namespace in Content Manager.

Structured MDL	Description
SecurityObject 11531 **'CAMID(":Authors")'** SecurityObjectDisplayName "Authors" SecurityObjectType **SecurityType_Role** **CustomViewList 11529 EndList**	**SecurityObject** is the imported security object from the last defined namespace. **'CAMID(":Authors")'** is the security control mechanism identifier for the security object from Content Manager. **SecurityType_Role** indicates that the type of security object from Content Manager is a role. Possible values are `SecurityType_Role`, `SecurityType_Group`, and `SecurityType_User`. **CustomViewList 11529 EndList** is a list of the custom views to which this security object is assigned. References to custom views may be unique identifiers or names.
SecurityObject 11539 'CAMID(":Analysis Users")' SecurityObjectDisplayName "Analysis Users" SecurityObjectType SecurityType_Role CustomViewList 11537 EndList	
SecurityNameSpace 11563 "GOnamespace" SecurityNameSpaceCAMID 'CAMID ("GOnamespace")'	**SecurityNameSpace** defines a second namespace from which security objects are retrieved. When multiple namespaces are defined in a custom view, they are organized by namespace, followed by the security objects from that namespace.
SecurityObject 11561 'CAMID ("GOnamespace:r:authid=2589996611")' SecurityObjectDisplayName "Root User Class" SecurityObjectType SecurityType_Role CustomViewList 11537 EndList	

Structured MDL	Description

`Cube 11535 "Cube1"`
`EncryptedPW"DD32E203AA9AFC5C26233A515A4E83A1DD32E`
`203AA9AFC5C26233A515A4E83A1DD32E20""3AA9AFC5C2623`
`A515A4E83A1DD32E203AA9AFC5C26233A515A4E83A1DD32E2`
`03AA9AFC""5C26233A515A4E83A1DD32E203AA9AFC5C26233`
`A515A4E83A1DD32E203AA9AFC5C26233""A515A4E83A1DD32`
`E203AA9AFC5C26233A515A4E83A1DD32E203AA9AFC5C26233`
`A515A4E""83A1DD32E203AA9AFC5C26233A515A4E83A1DD32`
`E203AA9AFC5C26233A515A4E83A1DD3""2E203AA9AFC5C262`
`33A515A4E83A1DD32E203AA9AFC5C26233A515A4E83A1DD32`
`E203AA""9AFC5C26233A515A4E83A1DD32E203AA9AFC5C262`
`33A515A4E83A1DD32E203AA9AFC5C2""6233A515A4E83"`
`Status New CubeCreation On Optimize Default`
`ConsolidatedRecords 10000000 PartitionSize 500000`
`PassesNumber 5`
`Compress False`
`DatabaseInfo "Local;" IncrementalUpdate False`
`ServerCube False CubeStamp`
`0 CubeCycle 0`
`BlockParentTotals False Caching False`
`UseAlternateFileName False`
`DrillThrough False`
`EndList DataSourceSignon False PublishEnable True`
`PublishStatus`
`None DimensionView 149`
`"All Categories" DimensionView 195 "All Categories"`
`MeasureInclude`
`225 Yes`
	This section creates the PowerCube.

`PowerCubeCustomViewList` **`11537 EndList`**	Defines the custom views assigned to the PowerCube. The list includes only the lowest custom views in the hierarchies. All ancestors are also considered to be assigned to the cube.
`AllocationAdd Measure 225 Type Default`	After you add the required measures, you may want to specify how they will be allocated across the various dimensions in your model.

Example - MDL Model Using an IBM Cognos 8 Data Source in Verb MDL

This is the order in which you would create a typical model using verb MDL:

- Create the data sources in Transformer.
- Create the dimensions and key performance measures.

- Create the standard dimension views.
- Create dimension views and custom views.
- Add the namespace and security objects.
- Populate the model and create the PowerCube.

The following example shows the same model, created using an IBM Cognos 8 data source, in verb MDL format.

Note: The `Add` form illustrates the verb that you use to define an object, to differentiate the syntax from that which Transformer uses when defining objects (the `Make` form). However, either form is acceptable. Also, you will note that each statement starts with the verb, followed by a lengthy series of options. For readability, you may wish to insert carriage returns between the keywords.

Verb MDL	Description
Make 103 **"GO Sales and Retailers" SourceType** Package **SourcePath** "/content/package[@name='GOSales and Retailers']" **PackageTimeStamp** "/content/package[@name='GOSales and Retailers']/model[@name='model']"	**Make** is the verb used to create a package or report. **"Go Sales and Retailers"** is the name of the IBM Cognos 8 package. All packages and queries must have unique names. **SourceType** defines the type of IBM Cognos 8 data source. The type can be Package, Report, or Cognos-SourceQuery. In this example, `SourceType` defines a package data source type. **SourcePath** is the path to the IBM Cognos 8 package stored in Content Manager. **PackageTimeStamp** is the version of the package last used in the model.
Make 11543 "QuantityReport" **SourceType** Report SourcePath "/content/package[@name='GO Sales and Retailers']/report[@name='QuantityReport']" PackageTimeStamp "/content/package[@name='GO Sales and Retailers']/report[@name='QuantityReport']"	**Make** defines a second IBM Cognos 8 package or report data source in the model. In this example, the **SourceType** defines a report data source type.

Verb MDL	Description
DataSourceMake 105 "GO Sales and Retailers~1" Separator "," SourceType CognosSourceQuery CharacterSet Default DecimalSep "." Thousandsep "," Columns True Timing PopYesCreateDefault **PackageReportSource 103 "GOSales and Retailers"** AutoSummary True SetCurrent True ServerSource False Speed False Presummarized False	**DataSourceMake** defines a query defined against a package or report SourceType. **PackageReportSource 103 "GO Sales and Retailers"** is the reference to the package on which this query is defined.
ColumnMake 107 **"[gosales_goretailers].[Orders].[Order number]"** DataSource 105 Origin Source Offset 0 Column "Order number" Storage Float64 Scale 0 Size 4 Decimals 0 InputScale 0 TimeArray Off Rollup CountAll	**"[gosales_goretailers].[Orders].[Order number]"** is the reference within the package or report to the query item or measure on which this column is defined. Following are more references to query items or measures on which the columns are defined.
ColumnMake 109 "[gosales_goretailers].[Orders]. [Retailer name]" DataSource 105 Origin Source Offset 1 Column "Retailer name" Storage Text Scale 0 Size 102 Decimals 0 Class Description InputScale 0 TimeArray Off	
ColumnMake 111 "[gosales_goretailers].[Orders]. [Order date]" DataSource 105 Origin Source Offset 2 Column "Order date" Storage Int32 Scale 0 Size 12 Decimals 0 Class Date InputScale 0 TimeArray Off	
ColumnMake 113 "[gosales_goretailers].[Orders]. [Order method]" DataSource 105 Origin Source Offset 3 Column "Order method" Storage Text Scale 0 Size 102 Decimals 0 Class Description InputScale 0 TimeArray Off	
ColumnMake 115 "[gosales_goretailers].[Orders]. [Order method code]" DataSource 105 Origin Source Offset 4 Column "Order method code" Storage Float64 Scale 0 Size 4 Decimals 0 InputScale 0 TimeArray Off Rollup CountAll	

Verb MDL	Description

`ColumnMake 117 "[gosales_goretailers].[Orders].[Product name]" DataSource 105 Origin Source Offset 5 Column "Product name" Storage Text Scale 0 Size 102 Decimals 0 Class Description InputScale 0 TimeArray Off`
 	
`ColumnMake 119 "[gosales_goretailers].[Orders].[Quantity]" DataSource 105 Origin Source Offset 6 Column "Quantity" Storage Float64 Scale 0 Size 2 Decimals 0 Class Quantity InputScale 0 TimeArray Off Rollup Sum FilterMake 11553 "Americas" FilterRef "[gosales_goretailers].[Americas]"`	

`FilterMake 11553 "Americas" FilterRef "[gosales_goretailers].[Americas]"`	**`FilterMake 115533 "Americas"`** selects a filter from the data source and imports it into the query. **`FilterRef "[gosales_goretailers].[Americas]"`** is the reference within the package or report to the filter on which this column is defined.
`DataSourceMake 11545 "QuantityReport~1" Separator "," SourceType CognosSourceQuery CharacterSet Default DecimalSep "." Thousandsep "," Columns True Timing PopYesCreateDefault PackageReportSource 11543 "QuantityReport" AutoSummary False SetCurrent True ServerSource False Speed False Presummarized False`  `ColumnMake 11547 "[Report].[Query1.0].[Product number]" Origin Source Offset 0 Column "Product number" Storage Float64 Scale 0 Size 1 Decimals 0 Class Quantity InputScale 0 TimeArray Off`  `ColumnMake 11549 "[Report].[Query1.0].[Production cost]" Origin Source Offset 1 Column "Production cost" Storage Float64 Scale 2 Size 1 Decimals 2 Class Quantity InputScale 0 TimeArray Off`	**`PackageReportSource 11543 "QuantityReport"`** is a second query, defined against a second package. Following are more references to query items or measures on which the columns are defined.

Verb MDL	Description
`DimMake 149 "Order date" DimType Date EarliestDate 19010101 LatestDate 21001231 ManualPeriods False DaysInWeek 127 NewCatsLock False ExcludeAutoPartitioning False DimDefaultCategory 0`	
`RootCatMake` `153 "Order date" Dimension 149 Inclusion Generate Lastuse 20070910 Date 0 Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False` **`DrillCatMake`** `155 "By Order date" Dimension 149 Root 153 Inclusion Suppress Filtered False Suppressed True PrimaryDrill True HideValue False YearBegins 20070101 PartialWeek Split ExtraWeek None WeekBegins Sunday LevelMake 161 "Year" Drill 155 Parent 0 Blanks "(blank)" Inclusion Generate DateFunction Year Generate Need RefreshLabel False RefreshDescription False`	When you create any dimension using MDL, you must manually create both the root category and drill category for that dimension. The verbs **`RootCatMake`** and **`DrillCatMake`** are used in this example because there is no `Add` verb form for these objects. Also, you must set the inclusion property to `Suppress` for the drill category so that it does not appear in the reporting components.
`RefreshShortName False NewCatsLock False CatLabFormat "YYYY" Timerank 10` **`UniqueCategories True`** `UniqueMove False Associations 163 "Order date" AssociationType Type_Query AssociationRole Role_Source AssociationReferenced "Order date" Associations 165 "Order date" AssociationContext 155 AssociationType Type_Query AssociationRole Role_OrderBy AssociationReferenced "Order date" SortOrder Int16 SortAs Ascending`	**`UniqueCategories True`** indicates that the Year level is unique. This is necessary because the level is associated with columns from multiple queries.

Verb MDL	Description

```
LevelMake 167 "Quarter" Drill 155
Parent 161
Blanks "( blank )" Inclusion Generate DateFunction
Quarter Generate
All RefreshLabel False
RefreshDescription False RefreshShortName False
NewCatsLock False
CatLabFormat 'YYYY
"Q" Q' Timerank 20 UniqueCategories True UniqueMove
False Associations
169 "Order date"
AssociationType Type_Query AssociationRole Role_Source
 AssociationReferenced
"Order date"
Associations 171 "Order date" AssociationContext 155
 AssociationType
Type_Query
AssociationRole Role_OrderBy AssociationReferenced
"Order date"
SortOrder Int16
SortAs Ascending


CatMake 233 "20040101-20041231"
Dimension 149 Drill 155
Levels 161 Parent 155 OrderBy Drill 155 Value "2004"
 Label "2004"
Lastuse 20070910
SourceValue "2004" Date 20040101 Filtered False
Suppressed False
Sign False HideValue False
IsKeyOrphanage False IsTruncated False Blanks False

CatMake 235 "20040101-20040331" Dimension 149 Drill
155
Levels 167 Parent 233 OrderBy Drill 155 Value
"20040101" Label "2004
Q 1" Lastuse 20070910
SourceValue "20040101" Date 20040101 Filtered False
Suppressed False
Sign False HideValue False
IsKeyOrphanage False IsTruncated False Blanks False


CatMake 243 "20040401-20040630"
Dimension 149 Drill 155
Levels 167 Parent 233 OrderBy Drill 155 Value
"20040401" Label "2004
Q 2" Lastuse 20070910
SourceValue "20040401" Date 20040401 Filtered False
Suppressed False
Sign False
HideValueFalse IsKeyOrphanage False IsTruncated False
 Blanks False
```

Verb MDL	Description

```
CatMake 251 "20040701-20040930"
Dimension 149 Drill 155
Levels 167 Parent 233 OrderBy Drill 155 Value
"20040701" Label "2004
Q 3" Lastuse 20070910
SourceValue "20040701" Date 20040701 Filtered False
Suppressed False
Sign False
HideValue False IsKeyOrphanage False IsTruncated False
 Blanks False

CatMake 259 "20041001-20041231"
Dimension 149 Drill 155
 Levels 167 Parent 233 OrderBy Drill 155 Value
"20041001" Label
"2004 Q 4" Lastuse 20070910
 SourceValue "20041001" Date 20041001 Filtered False
 Suppressed
False Sign False
 HideValue False IsKeyOrphanage False IsTruncated
False Blanks False

CatMake 581 "20050101-20051231"
Dimension 149 Drill 155
Levels 161 Parent 155 OrderBy Drill 155 Value "2005"
 Label "2005"
Lastuse 20070910
SourceValue "2005" Date 20050101 Filtered False
Suppressed False
Sign False
HideValue False IsKeyOrphanage False IsTruncated False
 Blanks False

CatMake 583 "20050101-20050331" Dimension 149 Drill
155
Levels 167 Parent 581 OrderBy Drill 155 Value
"20050101" Label "2005
Q 1" Lastuse 20070910
SourceValue "20050101" Date 20050101 Filtered False
Suppressed False
Sign False
HideValue False IsKeyOrphanage False IsTruncated False
 Blanks False

CatMake 591 "20050401-20050630"
Dimension 149 Drill 155
Levels 167 Parent 581 OrderBy Drill 155 Value
"20050401" Label "2005
Q 2"
Lastuse 20070910 SourceValue "20050401" Date 20050401
 Filtered False
Suppressed False Sign False HideValue False
IsKeyOrphanage False
IsTruncated False Blanks False
```

Verb MDL	Description

```
CatMake 599 "20050701-20050930"
Dimension 149 Drill 155 Levels 167 Parent 581 OrderBy
 Drill 155
Value "20050701"
Label "2005 Q 3" Lastuse 20070910 SourceValue
"20050701" Date 20050701
Filtered False Suppressed False Sign False HideValue
 False IsKeyOrphanage
False
IsTruncated False Blanks False

CatMake 607 "20051001-20051231"
Dimension 149 Drill 155 Levels 167 Parent 581 OrderBy
 Drill 155
Value "20051001" Label "2005 Q 4" Lastuse 20070910
SourceValue "20051001" Date 20051001 Filtered False
Suppressed False Sign False HideValue False
IsKeyOrphanage False
IsTruncated False Blanks False

CatMake 683 "20060101-20061231"
Dimension 149 Drill 155 Levels 161 Parent 155 OrderBy
 Drill 155
Value "2006" Label "2006" Lastuse 20070910 SourceValue
 "2006"
Date 20060101 Filtered False Suppressed False Sign
False
HideValue False IsKeyOrphanage False IsTruncated False
 Blanks False

CatMake 685 "20060101-20060331"
Dimension 149 Drill 155 Levels 167 Parent 683 OrderBy
 Drill 155
Value "20060101" Label "2006 Q 1" Lastuse 20070910
SourceValue "20060101" Date 20060101 Filtered False
Suppressed False Sign False HideValue False
IsKeyOrphanage False IsTruncated False Blanks False

CatMake 693 "20060401-20060630"
Dimension 149 Drill 155 Levels 167 Parent 683 OrderBy
 Drill 155
Value "20060401" Label "2006 Q 2" Lastuse 20070910
SourceValue "20060401" Date 20060401 Filtered False
Suppressed False Sign False HideValue False
IsKeyOrphanage False IsTruncated False Blanks False

CatMake 701 "20060701-20060930"
Dimension 149 Drill 155 Levels 167 Parent 683 OrderBy
 Drill 155
Value "20060701" Label "2006 Q 3" Lastuse 20070910
SourceValue "20060701" Date 20060701 Filtered False
Suppressed False Sign False HideValue False
IsKeyOrphanage False IsTruncated False Blanks False

CatMake 709 "20061001-20061231"
Dimension 149 Drill 155 Levels 167 Parent 683 OrderBy
 Drill 155
Value "20061001" Label "2006 Q 4" Lastuse 20070910
SourceValue "20061001" Date 20061001 Filtered False
Suppressed False Sign False HideValue False
IsKeyOrphanage False
IsTruncated False Blanks False
```

Verb MDL	Description

```
ViewMake 157 "All Categories" Dimension
149
Type All ViewCustomView 0

ViewMake 159 "Omit Dimension" Dimension 149
Type Omit ViewCustomView 0


DimMake 195 "Order method" DimType
Regular
NewCatsLock False ExcludeAutoPartitioning False
DimDefaultCategory
0


RootCatMake 197 "Order method" Dimension
195
Inclusion Generate Lastuse 20070910 Filtered False
Suppressed False
Sign False
HideValue False IsKeyOrphanage False IsTruncated False
 Blanks False

DrillCatMake 199 "By Order method" Dimension 195
Root 197 Inclusion Suppress Filtered False Suppressed
 True PrimaryDrill
True HideValue False

LevelMake 205 "Order method" Drill 199
Parent 0 Blanks "( Blank )" DateFunction None Generate
 None RefreshLabel
False
RefreshDescription False RefreshShortName False
NewCatsLock False
Timerank 0
UniqueCategories False UniqueMove False Associations
 207 "Order
method code"
AssociationType Type_Query AssociationRole Role_Source
 AssociationReferenced
 "Order method code" Associations 209 "Order method"
 AssociationType
 Type_Query AssociationRole Role_Label
AssociationReferenced "Order
method"


CatMake 267 "7" Dimension 195 Drill
199
Levels 205 Parent 199 Label "Sales visit" Lastuse
20070910
SourceValue "7" Filtered False Suppressed False Sign
 False HideValue
False
IsKeyOrphanage False IsTruncated False Blanks False


CatMake 285 "4" Dimension 195 Drill
199
Levels 205 Parent 199 Label "E-mail" Lastuse 20070910

SourceValue "4" Filtered False Suppressed False Sign
 False HideValue
False
IsKeyOrphanage False IsTruncated False Blanks False
```

Verb MDL	Description

`CatMake 309 "5" Dimension 195 Drill 199 Levels 205 Parent 199 Label "Web" Lastuse 20070910 SourceValue "5"Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False`
 	
`CatMake 319 "2" Dimension 195 Drill 199 Levels 205 Parent 199 Label "Telephone" Lastuse 20070910 SourceValue "2" Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False`
 	
`CatMake 357 "1" Dimension 195 Drill 199 Levels 205 Parent 199 Label "Fax" Lastuse 20070910 SourceValue "1" Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False`
 	
`CatMake 437 "3" Dimension 195 Drill 199 Levels 205 Parent 199 Label "Mail" Lastuse 20070910 SourceValue "3" Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False`
 	
`CatMake 475 "8" Dimension 195 Drill 199 Levels 205 Parent 199 Label "Special" Lastuse 20070910 SourceValue "8" Current Filtered False Suppressed False Sign False HideValue False IsKeyOrphanage False IsTruncated False Blanks False`	

`ViewMake 201 "All Categories" Dimension 195 Type All ViewCustomView 0` **`ViewMake 203 "Omit Dimension" Dimension 195 Type Omit ViewCustomView 0`**	These are the two standard views created for all dimensions. If you do not manually create the two default dimension views required for each dimension (**All Categories** and **Omit Dimension**), you must use the `ModelEnsureCompleteness` script to have Transformer create them automatically.

Verb MDL	Description
ViewMake 11555 "Authors~UserView" Dimension 195 **ViewCustomView 11529 Apex 197 Filter 319**	**ViewMake 11555 "Authors~User View"** creates a customized dimension view. **ViewCustomView 11529** is a reference to the view that this custom view uses in the dimension. **Apex 197 Filter 319** is a list of the categories that have customization options set, such as apexing or cloaking.
ViewMake 11557 "Sub Authors~User View" Dimension 195 ViewCustomView 11537 Apex 197 Filter 319	
MeasureMake 225 "Quantity" Rollup Sum IgnoreMissingValue False Storage Float64OutPutScale 0 Decimals 0 ReverseSign False IsCurrency False IsFolder False DrillThrough False EndList Associations 227 "Quantity" AssociationType Type_Query AssociationRole Role_Source AssociationReferenced "Quantity"	
CustomViewMake 11529 **"Authors"** **DimensionView 149 "All Categories"** **DimensionView 195 "Authors~User View"** **MeasureInclude 225 Yes**	**CustomViewMake** is the verb used to define a custom view. **"Authors"** is the name of the custom view. Custom view names must be unique. **DimensionView 149 "All Categories"** is a reference to a dimension (**149**) and the view that this custom view uses in that dimension, including all categories in the dimension. The second dimension, **DimensionView 195 "Authors~User View"**, is the customized dimension view used for this custom view. **MeasureInclude 225 Yes** indicates that measure **225** is included in the custom view. **MeasureInclude** values are **Yes** or **No**.

Verb MDL	Description

`CustomViewMake 11537 "Sub Authors"`
`DimensionView 149 "All Categories" DimensionView 195`
 	
`"Sub Authors~User View"`
	This entry creates a second custom view.

`MeasureInclude 225 No`	**`MeasureInclude 225 No`** indicates that measure **`225`** has been excluded from this custom view.

`CustomViewChildListUpdate`
`11529 StartList 11537 EndList` `CustomViewChildList`
`11537`
`StartList EndList`
	`CustomViewChildListUpdate` is the verb used to define the list of descendent custom views. There is only an Update version of this verb.
	`CustomViewChildListUpdate 11529 StartList 11537 EndList` defines the custom views that are descendents of the custom view "Authors".
	References to custom views may be unique identifiers or names.

`SecurityNameSpaceMake` **`11533 "Cognos" SecurityNameSpaceCAMID 'CAMID(":")'`**	**`SecurityNameSpace 11533 "Cognos" SecurityNameSpaceCAMID`** **`'CAMID(":")'`** is the namespace from which security objects are retrieved. In this example, **`"Cognos"`** is the name of the namespace in Content Manager.

Verb MDL	Description
SecurityObjectMake 11531 **'CAMID(":Authors")' SecurityNameSpace11533** SecurityObjectDisplayName "Authors" SecurityObjectType **SecurityType_Role CustomViewList11529 EndList**	**SecurityObjectMake** is the verb used to define a security object. **SecurityObject** is the imported security object from the last defined namespace. **'CAMID(":Authors")'** is the security control mechanism identifier for the security object from Content Manager. **SecurityNameSpace 11533** is the namespace that this security object comes from. **SecurityType_Role** indicates that the type of security object from Content Manager is a role. Possible values are SecurityType_Role, SecurityType_Group and SecurityType_User. **CustomViewList 11529 EndList** is a list of the custom views to which this security object is assigned. References to custom views may be unique identifiers or names.
SecurityObjectMake 11539 'CAMID(":Analysis Users")' SecurityNameSpace 11533 SecurityObjectDisplayName "Analysis Users" SecurityObjectType SecurityType_Role CustomViewList 11537 EndList	
SecurityNameSpaceMake 11563 "GOnamespace" SecurityNameSpaceCAMID 'CAMID ("GOnamespace")'	**SecurityNameSpaceMake** defines a second namespace from which security objects are retrieved. When multiple namespaces are defined in a custom view, they are organized by namespace, followed by the security objects from that namespace.
SecurityObjectMake 11561 'CAMID("GOnamespace:r:authid=2589996611")' SecurityNameSpace 11563 SecurityObjectDisplayName "Root User Class" SecurityObjectType SecurityType_Role CustomViewList 11537 EndList	

Verb MDL	Description
CubeMake 11535 "Cube1" EncryptedPW "DD32E203AA9AFC5C26233A515A4E83A1DD32E203AA9AFC5 C26233A515A4E83A1DD32E20""3AA9AFC5C26233A515A4E83 DD32E203AA9AFC5C26233A515A4E83A1DD32E203AA9AFC"" 5C26233A515A4E83A1DD32E203AA9AFC5C26233A515A4E83 A1DD32E203AA9AFC5C26233""A515A4E83A1DD32E203AA9A FC5C26233A515A4E83A1DD32E203AA9AFC5C26233A515A4 E""83A1DD32E203AA9AFC5C26233A515A4E83A1DD32E203A A9AFC5C26233A515A4E83A1DD3""2E203AA9AFC5C26233A 515A4E83A1DD32E203AA9AFC5C26233A515A4E83A1DD32E 203AA""9AFC5C26233A515A4E83A1DD32E203AA9AFC5C26 233A515A4E83A1DD32E203AA9AFC5C2" "6233A515A4E83" Status New CubeCreation On Optimize Default ConsolidatedRecords 10000000 PartitionSize 500000 PassesNumber 5 Compress False DatabaseInfo "Local;" IncrementalUpdate False ServerCube False CubeStamp 0 CubeCycle 0 BlockParentTotals False Caching False UseAlternateFileName False DrillThrough False EndList DataSourceSignon False PublisheEnable True PublishStatus None DimensionView 149 "All Categories" DimensionView 195 "All Categories" MeasureInclude 225 Yes	This section creates the PowerCube.
PowerCubeCustomViewListUpdate Cube 11535 StartList 11537 EndList	**PowerCubeCustomViewListUpdate** is the verb used to define the custom views assigned to the PowerCube. There is only an Update version of this verb. The list includes only the lowest custom views in the hierarchies. All ancestors are also considered to be assigned to the cube.
AllocationAdd Measure 225 Type Default	After you add the required measures, you may want to specify how they will be allocated across the various dimensions in your model.

Sample Scripts

The following sample scripts are included for use as a starting point. You can modify them to perform some of the more common MDL scripting operations.

Change Data Source Types Sample

This script changes a source file. It specifies the new file type (Excel) and the new file (Products.xls), and saves the model.

```
OpenPy "model.pyj" DataSourceUpdate
"Products (CSV)" SourceType ExcelCrossTab Source
"Products.xls" SavePy "model.pyj"
```

Change Data Source Sample

This script removes all categories in the Locations dimension and changes the source for this dimension to Locations.mdb, a Microsoft Access file. It then generates categories and creates the .mdc file. It does not save the changes to the model.

```
OpenMDL "c:\Outdoors.mdl"
CleanHouse Dimension "Locations" 20011231
DataSourceUpdate "Locations (CSV)" SourceType Access Source "c:\Locations.mdb"
 DataRange
"Locations"
PopulateFromQueries "Locations (CSV)"
CreateFiles
```

Update Selected Cubes Sample

This script deletes the measure Product Cost from two cubes and then updates the .mdc files for those cubes. It does not save the changes to the model.

```
OpenMDL "c:\Great Outdoors (Admin).mdl"
CubeUpdate "Americas" MeasureInclude "Product Cost"
No CubeUpdate "Europe" MeasureInclude "Product Cost" No
CreateFromCubes "Americas" "Europe"
```

Update All Cubes Sample

This script deletes all the categories in the Channels dimension, changes the data source for Channels to Channels.dbf, and generates categories from the new source. It then creates the .mdc file. It does not save the changes to the model.

```
OpenMDL "c:\Outdoors.mdl"
CleanHouse Dimension "Channels" 20011231 DataSourceUpdate
"Channels (dBase 4.0)" Source "c:\Channels.dbf"
PopulateFromQueries "Channels (dBase 4.0)" CreateFiles
```

Add a Description Sample

Every Transformer object can have a description up to 4,095 characters in length. However, in MDL the description must be divided into a series of smaller 256-character blocks, and all descriptions must be enclosed in quotation marks.

This example provides a description for the Products dimension and saves the model.

```
OpenMDL "c:\Outdoors.mdl"
DimUpdate "Products" DimInfo "This is
an example of a
'description' containing quotes." SaveMDL "c:\Outdoors.mdl"
```

Add Several Descriptions Sample

This script adds or changes a description for several objects and saves the model.

```
OpenMDL "c:\Outdoors.mdl" DimUpdate
"Products" DimInfo "Updating dimension description" ColumnUpdate "Product Line"
ColumnInfo "Updating column description"
LevelUpdate "Product Line" LevelInfo
"Updating level description" SaveMDL "c:\Outdoors.mdl"
```

Change a Cube Output Location Sample

This script changes the output location of the national cube to c:\mdcdir\national.mdc, and saves the model.

```
OpenPy "model.pyj" CubeUpdate "national"
MDCFile "c:\mdcdir\national.mdc" SavePy "model.pyj"
```

Change Several Cube Output Locations Sample

This script changes the output location of several cubes and generates the .mdc files in the new location. It does not save the changes in the model.

```
OpenMDL "c:\Great Outdoors (Admin).mdl"
CubeUpdate "Great Outdoors Sales" MDCFile "c:\Great Outdoors"
CubeUpdate "All Regions" MDCFile "c:\All Regions"
CubeUpdate "Americas" MDCFile "c:\Americas"
CubeUpdate "Far East" MDCFile "c:\Far East"
CubeUpdate "Europe" MDCFile "c:\Europe"
CreateFiles
```

Change the PowerCube Output Type Sample

This script changes the file type of a PowerCube and saves the model.

```
OpenPy "model.pyj"
SignonMake "name" PromptForPassword False UserId "user" Password "mypassword"
PowerCubeUpdate "national" DatabaseInfo "Local;;"
" Database "name" SavePy "model.pyj"
```

Change Optimization Setting Sample

This script changes the optimization setting of several cubes to use the older `Categories` option in cases where autopartitioning is either not supported or not desired. It then generates the .mdc files without saving the changes.

```
OpenMDL "c:\Great Outdoors (Admin).mdl"
CubeUpdate "Great Outdoors Sales" Optimize Categories
CubeUpdate "All Regions" Optimize Categories
CubeUpdate "Americas" Optimize Categories
CubeUpdate "Far East" Optimize Categories
CubeUpdate "Europe" Optimize Categories
CreateFiles
```

Automate CleanHouse Sample

This script uses the MDL verb `CleanHouse`, which is equivalent to selecting the Clean House command on the Tools menu of the Windows interface, followed by a date in the format `YYYYMMDD`. This command removes all categories with a last use date less than the specified date.

Tip: To remove all categories in the model, specify a future date.

This script removes all categories in the Product dimension that have a last use date prior to November 1, 2006, and saves the model.

```
OpenPy "model.pyj"
CleanHouse Dimension "Product" 20061101 SaveMDL "model.mdl"
```

Convert Model File Formats Sample

This script converts a legacy format (*.def) file into .mdl and then .pyj format.

```
OpenDef "\customer\defdat\01xbudgt.def"
SaveMDL "\customer\defdat\01xbudgt.mdl"
SavePy "\customer\defdat\01xbudgt.pyj"
```

Create Cubes Based on Dimension Views Sample

This script creates four dimension views, creates four PowerCubes based on the dimension views, and then creates the .mdc files. Each cube relates to one region, except for `All Regions` which has a drill through to all regions.

```
OpenMDL "c:\Outdoors.mdl"
ViewMake "Region Summary" Dimension "Locations" LevelSummary
Dimension "Locations" Drill "By Region" Levels "Region"Apex
"Region"
ViewMake "North America" Dimension "Locations"
Apex "Region"Summary "Europe"Summary "Far East"
ViewMake "Far East" Dimension "Locations"
Apex "Region"Summary "Europe"Summary "North America"
ViewMake "Europe" Dimension "Locations"
Apex "Region" Summary" Far East" Summary "North
America"
CubeMake "All Regions" MdcFile "c:\All Regions.mdc"
DrillThrough True "c:\NorthAmerica.mdc"
"" "c:\europe.mdc" "" "c:\fareast.mdc" "" EndList
DimensionView "Locations""Region Summary"
CubeMake "North America" MdcFile "c:\North America.mdc"
DimensionView "Locations""North America"
CubeMake "Far East" MdcFile "c:\Far East.mdc"
DimensionView "Locations" "Far East"
CubeMake "Europe" MdcFile "c:\Europe.mdc"
DimensionView "Locations" "Europe"
CreateFiles
```

Update Conversion Rates Sample

This script adds or changes the currency rates for German marks and French francs for November and December, 2006; that is, before these countries entered the EMU.

```
OpenMDL "C:\Great Outdoors (Admin).mdl"
CurrencyUpdate "German Marks" Levels "Month" CurrencyRateList
EffectiveDate "20021101-20061130" ConversionRate 1.666
EffectiveDate "20021201-20061231" ConversionRate 1.777 EndList
CurrencyUpdate "French Francs" Levels "Month" CurrencyRateList
```

```
EffectiveDate "20021101-20061130" ConversionRate 1.666
EffectiveDate "20021201-20061231" ConversionRate 1.777 EndList
```

Disable Incremental Update Sample

This script disables incremental update for a cube and generates the .mdc file for that cube.

```
OpenMDL "C:\Great Outdoors (Admin).mdl"
CubeUpdate "Europe" IncrementalUpdate False CreateFromCubes "Europe"
```

Chapter 4: Syntax Conventions

The Transformer *Developer Guide* uses the following syntax and naming conventions:

- MDL verbs and keywords appear in `code` or **bold** form. Verbs and keywords are not case-sensitive, but otherwise must be typed as is. For example, if the syntax includes `CatMake`, you can type `CatMake`, `catmake`, or `CATMAKE`.
- *Italicized* words are placeholders for things you must supply. If more than one placeholder can be supplied, the italicized word is plural, such as *`objCats`* in the following example:
  `StartList` *`objCats`* `EndList`
- Italicized words starting with *obj* are placeholders to identify an object. The identification must be an object name, object identifier, or both. For example, *`objCat`* is a placeholder for a category and might appear in the MDL file as `135 "Dishwashers"`, which identifies a category with object identifier `135` and object name `Dishwashers`.
- Square brackets [ ] indicate optional items. If only one of two or more optional items may be chosen, they are separated by a pipe (|). For example, `[red | green]` means that you can specify red or green, or neither. The syntax, `[red][green]` means that you can use either, neither, or both red and green.
- Braces { } indicate a set of choices, separated by a pipe, from which you must choose one. For example, `{red|green}` means that you must specify either red or green, but not both.

When the syntax is too long to fit on one line, it wraps at a space between characters.

Data Type Conventions

The Transformer *Developer Guide* uses the following data type conventions.

Data type	Description
`bignum`	32-bit number between -2,147,483,648 and 2,147,483,647
`number`	16-bit number between -32,768 and 32,767
`uns`	unsigned number
`string`	character string, enclosed in single or double quotes, up to 256 characters in length

Syntax Example

The following example uses the MDL verb `CatMake`. You must type this verb and then the identification for a category on which it operates, but all other parameters are optional, as indicated by their enclosing square brackets.

```
CatMakeobjCat [Dimension
objDim] [DrillobjDrillCat][Levels
objLevel] [ParentobjLevel][
catopts] [ParentListobjLevels]EndList
```

The following more complex example is based on the MDL keyword `AllocationAdd`. Options include `Dimension`, which must be followed by the dimension identifier, and `Drill`, which must be followed by a drill category identifier.

These mandatory parameters may be followed by one or more optional arguments:

- `Levels`, followed by the identification of a level
- `Category`, followed by the identification of a root category
- `Category`, followed by the identification of a regular category

You then type `Measure` followed by the identification of a measure, and `Type` followed by the measure type. Note that *type* does not have the prefix *obj*. This indicates that it is not a Transformer object.

Finally, you can optionally include `AllocationMeasure` and the identification of a measure.

```
AllocationAdd [DimensionobjDim][Drill
objDrillCat] [LevelsobjLevel
|CategoryobjRootCat|CategoryobjCat]MeasureobjMeasureTypetype
[AllocationMeasureobjMeasure
```

Syntax Requirements

You can have up to 256 characters on one line in an MDL file. You can include carriage returns between keywords to reduce the length of a line.

To create a remark or temporarily remove a line from an MDL file, type two slash marks (`//`) immediately before the text you want to remove or create as a remark.

Chapter 5: MDL Verbs

This section of the *Developer Guide* provides an alphabetical list of all the MDL verbs that you can use with IBM Cognos Transformer.

Before you begin, make sure you review "Syntax Conventions" and the topic that explains when to use object identifiers or category codes: "Locating Objects Uniquely".

Verb Types

Most objects in MDL have four verb types associated with them: `Add`, `Delete`, `Make`, and `Update`; for example, `CatAdd`, `CatDelete`, `CatMake`, and `CatUpdate`.

- `Add` verbs add the object to the model. An error is issued if the object already exists. Examples for this verb type show you how to use it to create an object and specify its required settings.
- `Delete` verbs remove the object from the model. An error is issued if the object does not exist.
- `Make` verbs add the object if it does not already exist, and update the object if it does exist. `Make` verbs combine the functionality of both the `Add` and `Update` verbs. Examples for this verb type show the object definitions that are generated by Transformer when you save a model as an .mdl file.
- `Update` verbs update existing objects. An error is issued if the object does not exist.

Below is a list of the Transformer objects and the verbs you can use with each one.

Transformer objects	Associated verbs
Model	`NewModel`
DataSource	`DataSourceAdd`, `DataSourceDelete`, `DataSourceMake`, `DataSourceUpdate`, `SourceListUpdate`, `CognosPackageAdd`, `CognosPackageDelete`, `CognosPackageMake`, `CognosPackageUpdate`
Filter	`CognosPackageFilterAdd`, `CognosPackageFilterMake`, `CognosPackageFilterDelete`, `CognosPackageFilterUpdate`
Prompt	`PromptAdd`, `PromptDelete`, `PromptMake`, `PromptUpdate`

Transformer objects	Associated verbs
Column	ColumnAdd, ColumnDelete, ColumnMake, ColumnUpdate, ColumnListUpdate
Dimension	DimAdd, DimDelete, DimMake, DimUpdate, DimensionListUpdate
Subdimension	SubDimRootMake, SubDimRootUpdate
Root Category	RootCatMake, RootCatUpdate
Drill Category	DrillCatMake
Level	LevelAdd, LevelDelete, LevelMake, LevelMoveAfter, LevelMoveBefore, LevelNewDrill, LevelUpdate
Category	CatAdd, CatDelete, CatJoin, CatMake, CatMorph, CatMoveLevel, CatMoveVertical, CatUpdate, FilterCat
Special Category	SpecialCatAdd, SpecialCatDelete, SpecialCatMake, SpecialCatUpdate
View	ApexCat, ViewAdd, ViewDelete, ViewMake, ViewUpdate, ViewListUpdate
Custom View	CustomViewAdd, CustomViewMake, CustomViewDelete, CustomViewUpdate, CustomViewChildListUpdate, PowerCubeCustomViewListUpdate, SecurityNamespaceAdd, SecurityNamespaceDelete, SecurityNamespaceMake, SecurityNamespaceUpdate, SecurityObjectAdd, SecurityObjectDelete, SecurityObjectMake, SecurityObjectUpdate,
Measure	MeasureAdd, MeasureDelete, MeasureMake, MeasureListUpdate, MeasureUpdate, AllocationAdd
PowerCube	CubeAdd, CubeDelete, CubeMake, CubeUpdate, PowerCubeDelete, PowerCubeListUpdate PowerCubeCustomViewListUpdate

Transformer objects	Associated verbs
PowerCube Group	CubeGroupAdd, CubeGroupDelete, CubeGroupMake, CubeGroupUpdate, PowerCubeDelete, PowerCubeListUpdate
PowerCube Group Cube	CubeGroupCubeAdd, CubeGroupCubeDelete, CubeGroupCubeListUpdate, CubeGroupCubeMake, CubeGroupCubeUpdate
Signon	SignonAdd, SignonListUpdate, SignonMake, SignonUpdate, SignonDelete, CognosPackageDatasourceConnectionAdd, CognosPackageDatasourceConnectionMake, CognosPackageDatasourceConnectionUpdate, CognosPackageDatasourceConnectionDelete
Dimension Calculation Definition	DimCalcDefAdd, DimCalcDefDelete, DimCalcDefMake, DimCalcDefUpdate
Currency Table	CurrencyTableAdd, CurrencyTableDelete, CurrencyTableMake, CurrencyTableUpdate
Currency	CurrencyAdd, CurrencyDelete, CurrencyMake, CurrencyUpdate
Association	AssociationAdd, AssociationDelete, AssociationMake, AssociationUpdate, CurrencyTableAdd, CurrencyTable Make, CurrencyTableUpdate, DimensionAdd, DimensionMake, DimensionUpdate, LevelAdd, LevelMake, LevelUpdate, MeasureAdd, MeasureMake, MeasureUpdate

You can also use the following action verbs to manipulate Transformer objects.

- AutoDesign
- CatUpdateAll
- CleanHouse
- CreateColumns
- CreateFiles
- CreateFromCubes
- CreateFromQueries

- EventEnd
- EventStart
- MDCCheckServer
- MDCClear
- ModelEnsureCompleteness
- NewModel
- OpenDef
- OpenMDL
- OpenPY
- PopulateFromQueries
- PopulateModel
- ReportPartitions
- SaveMDL
- SavePY
- SummarizeCat
- SummarizeLevel
- UpdateForwardReference
- UpdatePowerCubes

AllocationAdd

The AllocationAdd verb allocates summary data to a level, dimension, or category.

Its equivalent on the Windows interface is the **Allocation** tab on the **Level**, **Dimension**, and **Category** property sheets.

Only one of level, root category, or category can be specified.

Every measure in a model must have an AllocationAdd statement that contains the name of the measure and the option Type Default. Transformer creates this statement if the measure is created on the Windows interface or if the verb ModelEnsureCompleteness is used. Otherwise, you must manually add the statement when you create a measure.

If a measure is allocated, then a second AllocationAdd statement defines the allocation.

Note: Category object names differ between MDL and the Windows interface. MDL uses the category code as the object name. For more information, see "Locating Objects Uniquely".

The syntax is as follows:

AllocationAdd [**Dimension** *objDim*][**Drill** *objDrillCat*][**Levels** *objLevel*|**Category** *objRootCat*|**Category** *objCat*] **Measure** *objMeasure* **Type** *type* [**AllocationMeasure** *objMeasure*]

Argument	Description
Dimension *objDim*	Specifies the level, root category, or category. *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Specifies the level, root category, or category. *objDrillCat* can be the object name, object identifier, or both.
Levels *objLevel*	Indicates that the allocation should be done to a level, and specifies the level. *objLevel* can be the object name, object identifier, or both. This argument is optional.
Category *objRootCat*	Indicates that the allocation should be done to a dimension, and specifies the root category of the dimension. *objRootCat* can be the object name, object identifier, or both. This argument is optional.
Category *objCat*	Indicates that the allocation should be done to a category, and specifies the category. *objCat* can be the object name, object identifier, or both. This argument is optional.
Measure *objMeasure*	Specifies the measure that is being allocated. The measure can be identified by object name, object identifier, or both. This argument is mandatory.
Type *type*	Specifies the type of allocation. *type* can be `Default`, `None`, `Constant`, or `Allocate`. The default is `Constant` if the allocation is to a dimension. For allocations to levels or categories it is `None`. This argument is mandatory.
AllocationMeasure *objMeasure*	Specifies the measure used as the basis for the allocation proportions. *objMeasure* can be the object name or object identifier. This argument is optional.

Example

This is an example of the `AllocationAdd` statement that Transformer requires for every measure.

```
AllocationAdd Measure "Revenue" Type Default
```

A second `AllocationAdd` statement exists for allocated measures. This example allocates forecast data in the Regions dimension (identified by its root category) according to the distribution of the Revenue measure.

```
AllocationAdd Category "Regions" Measure "Forecast" Type Allocate
AllocationMeasure "Revenue"
```

ApexCat

The `ApexCat` verb makes the selected category the highest category in a view. In PowerCubes that are created using this view, users will see only the apex category and its descendants.

To invoke the equivalent command on the Windows interface, open the left pane of the category viewer for the selected view and category and, from the **Diagram** menu, click **Apex**.

You cannot apply this verb to special categories.

Note: Category object names differ between MDL and the Windows interface. MDL uses the category code as the object name. For more information, see "Locating Objects Uniquely".

The syntax is as follows:

```
ApexCat objView [
Dimension objDim]{
Category objCat|
Category objRootCat|
Category objDrillCat}
```

Argument	Description
ApexCat *objView*	Specifies the dimension view or custom security view *objView* in which the category is to be apexed. *objView* can be the object name, object identifier, or both.
Dimension *objDim*	Specifies the view. *objDim* can be the object name, object identifier, or both.
Category *objCat*	Specifies a regular category as the apex. *objCat* can be the object name, object identifier, or both. One of the regular, root, or drill categories must be specified.
Category *objRootCat*	Specifies the root category as the apex, which has the effect of removing the apex. *objRootCat* can be the object name, object identifier, or both. One of the regular, root, or drill categories must be specified.

Argument	Description
Category *objDrillCat*	Specifies a drill category as the apex. *objDrillCat* can be the object name, object identifier, or both. One of the regular, root, or drill categories must be specified.

Example

This example applies the verb to the Outdoor Products category in the dimension view View1.

```
ApexCat "View1" Dimension "Products" Category "Outdoor Products"
```

AssociationAdd

The `AssociationAdd` verb adds an association to a measure, level, dimension, category, or currency conversion table.

Its Windows interface equivalent is adding an association on the property sheet for a measure, level, dimension, category, or currency conversion table.

The syntax is as follows:

`AssociationAdd` *`objAssoc assocopts`*

Argument	Description
AssociationAdd *objAssoc*	Specifies the name of the association *objAssoc* that you want to create and associate with an existing object. *objAssoc* can be the object name, object identifier, or both.
assocopts	Specifies the type and details of the association *objAssoc*. For the complete list of options, see assocopts.

Example

This is an example of an `AssociationAdd` statement.

```
LevelMake 2979 "Branch Code" Drill 2969
.
.
.
AssociationAdd 6391 "Branch Code" AssociationLevel 2979 AssociationType Type_
Query AssociationRole Role_Source AssociationReferenced "Branch Code"
```

AssociationDelete

The `AssociationDelete` verb removes the association from a measure, level, dimension, category, or currency conversion table.

Its Windows interface equivalent is deleting an association from the property sheet for a measure, level, dimension, category, or currency conversion table.

The syntax is as follows:

`AssociationDelete` *`objAssoc`*

Argument	Description
AssociationDelete *objAssoc*	Specifies the name of the association *objAssoc* whose association you want to remove from an existing object. *objAssoc* can be the object name, object identifier, or both.

Example

This is an example of an `AssociationDelete` statement.

```
LevelMake 2979 "Branch Code" Drill 2969
.
.
.
AssociationDelete 6391 "Branch Code"
```

AssociationMake

The `AssociationMake` verb creates an association or updates an existing one.

Its Windows interface equivalent is creating or updating an association on the property sheet for a measure, level, dimension, category, or currency conversion table.

The syntax is as follows:

`AssociationMake` *`objAssoc assocopts`*

Argument	Description
AssociationMake *objAssoc*	Specifies the name of the association *objAssoc* that you want to create or update and associate with an existing object. *objAssoc* can be the object name, object identifier, or both.
assocopts	Specifies the type and details of the association *objAssoc*. For the complete list of options, see assocopts.

Example

This is an example of an `AssociationMake` statement.

```
LevelMake 2979 "Branch Code" Drill 2969
.
.
.
AssociationMake 6391 "Branch Code" AssociationLevel 2979 AssociationType Type_
Query AssociationRole Role_Source AssociationReferenced "Branch Code"
```

AssociationUpdate

The `AssociationUpdate` verb updates an existing association.

Its Windows interface equivalent is updating an association on the property sheet for a measure, level, dimension, category, or currency conversion table.

The syntax is as follows:

`AssociationUpdate` *`objAssoc assocopts`*

Argument	Description
AssociationUpdate *objAssoc*	Specifies the name of an existing association *objAssoc* that you want to update. *objAssoc* can be the object name, object identifier, or both.
assocopts	Specifies the type and details of the association *objAssoc*. For the complete list of options, see assocopts.

Example

This is an example of an `AssociationUpdate` statement.

```
AssociationUpdate 6391 "Branch Code" AssociationLevel 2979 AssociationType Type_
Query AssociationRole Role_Source AssociationReferenced "Branch Code"
```

AutoDesign

The `AutoDesign` verb generates a preliminary model of dimensions and measures based on relationships that Transformer detects in a non-IBM Cognos 8 data source.

In order to use the `AutoDesign` verb, your model must contain at least one data source and one column.

On the Windows interface, you may choose to run **AutoDesign** from the **New Model** wizard or the **Tools** menu. You can also set it to run automatically on model creation from the **Preferences** dialog box.

The syntax, which does not require any parameters, is as follows:

AutoDesign

CatAdd

The `CatAdd` verb creates a category at the specified location in the model.

The equivalent on the Windows interface is to open the category viewer (diagram), click the right side of an existing category, and drag to the right to add a category at that same level.

The syntax is as follows:

```
CatAdd objCat[
Dimension objDim][
Drill objDrillCat]
Levels objLevel Parent objCat[
catopts][ParentList objCats EndList]
```

Argument	Description
CatAdd *objCat*	Creates the category *objCat*. *objCat* must be the object name and can include the object identifier.
Dimension *objDim*	Specifies a dimension for the category. *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Specifies a drill category for the category. *objDrillCat* can be the object name, object identifier, or both.
Levels *objLevel*	Specifies a level for the category. *objLevel* can be the object name or object identifier. This argument is mandatory.
Parent *objCat*	Specifies a parent category. *objCat* can be the object name or object identifier. This argument is mandatory.
catopts	Optional parameters that describe the category in greater detail. Some options are set by default if you do not set them. For the complete list of options, see catopts.
ParentList *objCats*	Specifies parents for the category if the category is at a convergence level. *objCats* is a list of one or more parent categories, each identified by object name or object identifier.

For more information about creating categories, see "CatMake" (p. 96).

Example

This is an example of a `CatAdd` statement that creates the Back Packs category:

```
CatAdd "Back Packs" Dimension "Products" Drill "By Product Line" Levels
"Product Type" Parent "Outdoor Products" SourceValue "Back Packs"
IsKeyOrphanage False IsTruncated False Blanks False
```

CatDelete

The `CatDelete` verb removes a category from the model.

The Windows interface equivalent is to select the category and, from the **Edit** menu, click **Delete**.

Note: Category object names differ between MDL and the Windows interface. MDL uses the category code as the object name. For more information, see "Locating Objects Uniquely".

The syntax is as follows:

```
CatDelete objCat [
 Dimension objDim]
```

Argument	Description
CatDelete *objCat*	Deletes the category *objCat*. *objCat* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the category. *objDim* can be the object name, object identifier, or both.

Example

This is an example of a `CatDelete` statement that removes the Germany category.

```
CatDelete "Germany" Dimension "Locations"
```

CatJoin

The `CatJoin` verb joins a category to its specified parent category. This verb is required when creating alternate drill-down paths, and is a useful means of making similar categories unique.

Note: Category object names differ between MDL and the Windows interface. MDL uses the category code as the object name. For more information, see "Locating Objects Uniquely". There is no Windows interface equivalent.

`CatJoin` is required when using alternate drill-down paths. It links a category in the convergence level with the parent in the alternate drill-down path.

The syntax is as follows:

```
CatJoin objCat[
Dimension objDim]
Parent objCat
```

Argument	Description
CatJoin *objCat*	Specifies the category *objCat* that is to be joined. *objCat* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the category. *objDim* can be the object name, object identifier, or both.
Parent *objCat*	Specifies the parent that is to be joined. *objCat* can be the object name or object identifier. This argument is mandatory.

Example

This is an example of a `CatJoin` statement that joins the category 1014 to its parent, Canada.

```
CatJoin "1014" Dimension "Channels" Parent "Canada"
```

CatMake

The `CatMake` verb creates a new category or updates an existing one.

The equivalent to create a category on the Windows interface is to open the category viewer (diagram), click the right side of an existing category, and drag to the right to add a category at that same level. The equivalent to modify an existing category is to change the settings on its property sheet, as required.

Category names cannot contain a tilde (~) or an at sign (@).

Note: Category object names differ between MDL and the Windows interface. MDL uses the category code as the object name. For more information, see "Locating Objects Uniquely".

The syntax is as follows:

```
CatMake objCat[
Dimension objDim][
Drill objDrillCat Levels[objLevel][
Parent objLevel][
catopts][ParentList objLevels EndList]
```

Argument	Description
CatMake *objCat*	Creates the category *objCat* or modifies it if it exists. *objCat* can be the object name, object identifier, or both. Include the object name if the category does not exist.

Argument	Description
Dimension *objDim*	When creating a category, this specifies its dimension. When updating a category, this is used, if necessary, to uniquely identify it. *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	When creating a category, this specifies its drill category. This argument is not necessary when updating a category. *objDrillCat* can be the object name, object identifier, or both.
Levels *objLevel*	When creating a category, this specifies its level. This argument is not necessary when updating a category. *objLevel* can be the object name or object identifier.
Parent *objCat*	When creating a category, this specifies its parent category. This argument is not necessary when updating a category. *objCat* can be the object name, or object identifier.
catopts	Optional parameters that describe the category in greater detail. Some options are set by default if you do not set them. For the complete list of options, see catopts. If the category exists, previously set options are retained unless you change them with this command.
ParentList *objCats*	When creating a category that is at a convergence level, this specifies its parents. When updating a category, this can be used to add one or more new parents. *objCats* is a list of one or more parent categories, each identified by object name or object identifier.

Example

This is an example of a `CatMake` statement that defines the category Back Packs.

```
CatMake "Back Packs" Dimension "Products" Drill "By Product Line" Levels
"Product Type" Parent "Outdoor Products" Lastuse 19971202 SourceValue "Back
Packs" Filtered False Suppressed False Sign False IsKeyOrphanage False
IsTruncated False Blanks False
```

CatMorph

The `CatMorph` verb changes the type of the specified category.

Note: Category object names differ between MDL and the Windows interface. MDL uses the category code as the object name. For more information, see "Locating Objects Uniquely". There is no Windows interface equivalent.

`CatMorph` exists so that MDL is a complete representation of a Transformer model when a category is changed. For example, when a leaf category acquires a child it changes (or "morphs") into a parent. This is an internal operation and when the model is saved as an .mdl file, the `CatMorph` statement does not appear in the file.

The syntax is as follows:

```
CatMorph objCat [
Dimension objDim]
CatType type
```

Argument	Description
CatMorph *objCat*	Specifies the category *objCat* that is to be changed. *objCat* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the category. *objDim* can be the object name, object identifier, or both.
CatType *type*	Specifies the category type. The type can be `Leaf`, `Special`, `Parent`, `Drill`, or `Root`. The type is mandatory.

CatMoveVertical

The `CatMoveVertical` verb moves a category to a new position within a level.

The Windows equivalent is to open the category viewer and drag the category from one position to another within a level.

This verb does not change the object identifier of the category. Rather, it determines how the category appears in the dimension viewer and its position in the .mdc file, which affects how it is viewed in the reporting components.

Note: Category object names differ between MDL and the Windows interface. MDL uses the category code as the object name. For more information, see "Locating Objects Uniquely".

The syntax is as follows:

```
CatMoveVertical objCat [
Dimension objDim]
Parent objLevel Sibling objCat
```

Argument	Description
CatMoveVertical *objCat*	Specifies the category that is to be moved. *objCat* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the category. *objDim* can be the object name, object identifier, or both.
Parent *objLevel*	Specifies the parent level of the category in its new position. The parent must be specified even if it remains the same. *objLevel* can be the object name or object identifier.
Sibling *objCat*	Specifies the sibling category that becomes the next category below the moved category. *objCat* can be the object name or object identifier. This argument is mandatory.

Example

This is an example of a `CatMoveVertical` statement that moves the category Sunblock to the position above the category Tents.

```
CatMoveVertical "Sunblock" Dimension "Products" Parent "Outdoor Products"
Sibling "Tents"
```

CatUpdate

The `CatUpdate` verb updates an existing category.

Its Windows interface equivalent is to modify the required settings on the **Category** property sheet.

For more information about updating categories, see "CatMake" (p. 96).

The syntax is as follows:

```
CatUpdate objCat [
Dimension objDim]
[catopts] [ParentList objCats EndList]
```

Argument	Description
CatUpdate *objCat*	Specifies the category to update. *objCat* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the category. *objDim* can be the object name, object identifier, or both.

Argument	Description
catopts	Optional parameters that describe the category in greater detail. For the complete list of options, see catopts. Previously set options are retained unless you change them with this command.
ParentList *objCats*	Specifies parents of the category. This can be used to add one or more new parents for the category. *objCats* are the added parent categories, each identified by its object name or object identifier.

CatUpdateAll

The `CatUpdateAll` verb changes the last use value of all categories in the model to the current date. There is no Windows interface equivalent that allows you to globally change this setting on the **General** tab of all **Category** property sheets.

The syntax is as follows:

`CatUpdateAll`

CleanHouse

The `CleanHouse` verb deletes categories with a last use date that is earlier than the date specified.

The equivalent on the Windows interface is the **Clean House** command on the **Tools** menu.

Each time a category is used, its last use value is set to the current system date. `LastUse` is a category option and can be found in `CatMake` statements in MDL or on the **Category** property sheet in the Windows interface.

The syntax is as follows:

`CleanHouse` `[`**`Dimension`** *`objDim`*`]` *`date`*

Argument	Description
Dimension *objDim*	Specifies a dimension within which the appropriate categories should be deleted. *objDim* can be the object name, object identifier, or both. If the dimension is not specified, all dimensions are affected.
date	Specifies the date, in the format `YYYYMMDD`. To delete all categories, specify a future date.

Example

This example removes all categories in the **Products** dimension with a **Last Use** date prior to 20070101 (January 1, 2007).

```
CleanHouse Dimension "Products" 20070101
```

CognosPackageAdd

The `CognosPackageAdd` verb creates a package or report data source definition in the model. The package or report definition is a reference to a package or report that is defined in IBM Cognos 8. You can then create a data source definition that has columns based on items that are selected from the package or report.

Notes:

- If you include a data source ID for the new package or report in the syntax, and the object already exists, you will receive an error message.
- To insert a dimension from a package (the **Insert Dimension From Package** command on the **Edit** menu, you must first add the package and data source and then add a dimension from the data source. For more information about adding a dimension from a data source, see "DimAdd" (p. 145).

The syntax for a query based on an IBM Cognos 8 package or report is as follows:

`CognosPackageAdd` *`objDataSource`* `[cognospackage_opts]`

Argument	Description
CognosPackageAdd *objDataSource*	Creates the data source *objDataSource*. *objDataSource* must be the object name and can include the object identifier.
cognospackage_opts	Optional parameters that can describe the package or report in greater detail. For the complete list of options, see "cognospackageopts" (p. 218).

Example

This example adds the IBM Cognos 8 package data source Go Sales and Retailers to the model.

```
CognosPackageMake 103 "GO Sales and Retailers" SourceType
Package SourcePath "/content/package[@name='GO Sales
and Retailers']" PackageTimeStamp "/content/package[@name='GO Sales
and Retailers']/model[@name='model']"
```

This example adds the IBM Cognos 8 report data base GSR_rpt_1 to the model.

```
CognosPackageMake 115 "GSR_rpt_1" SourceType Report SourcePath
"/content/package[@name='GO Sales and Retailers']/report[@name='GSR_rpt_1']"
PackageTimeStamp "/content/package[@name='GO
Sales and Retailers']/report[@name='GSR_rpt_1']"
```

CognosPackageDelete

The `CognosPackageDelete` verb removes a package or report data source from the model.

The Windows interface equivalent is to click **Delete** on the **Edit** menu when a data source is selected.

For more information about IBM Cognos 8 data sources, see the Transformer *User Guide* and the Framework Manager *User Guide*.

The syntax is as follows:

`CognosPackageDelete` *`objDataSource`*

Argument	Description
CognosPackageDelete *objDataSource*	Deletes the data source *objDataSource*. *objDataSource* can be the object name, object identifier, or both.

Example

This example deletes the All Staff Count (Excel) data source.

```
CognosPackageDelete "All Staff Count (Excel)"
```

CognosPackageMake

The `CognosPackageMake` verb creates a package or report data source definition in the model. If one already exists, it is updated. If one does not exist, it is added. The package or report definition is a reference to a package or report that is defined in IBM Cognos 8. You can then create a data source definition which has columns based on items that are selected from the package or report.

Note: To insert a dimension from a package (the **Insert Dimension From Package** command on the **Edit** menu, you must first add the package and data source and then add a dimension from the data source. For more information about adding a dimension from a data source, see "DimAdd" (p. 145).

The syntax for a query based on an IBM Cognos 8 package or report is as follows:

`CognosPackageMake` *`objDataSource`* `[cognospackage_opts]`

Argument	Description
CognosPackageMake *objDataSource*	Creates the data source *objDataSource*. *objDataSource* must be the object name and can include the object identifier.
cognospackage_opts	Optional parameters that can describe the package or report in greater detail. For the complete list of options, see "cognospackageopts" (p. 218).

Example

This example adds the IBM Cognos 8 package data source Go Sales and Retailers to the model.

```
CognosPackageMake 103 "GO Sales and Retailers" SourceType
Package SourcePath "/content/package[@name='GO Sales
and Retailers']" PackageTimeStamp "/content/package[@name='GO Sales
and Retailers']/model[@name='model']"
```

This example adds the IBM Cognos 8 report data base GSR_rpt_1 to the model.

```
CognosPackageMake 115 "GSR_rpt_1" SourceType Report SourcePath
"/content/package[@name='GO Sales and Retailers']/report[@name='GSR_rpt_1']"
PackageTimeStamp "/content/package[@name='GOSales
and Retailers']/report[@name='GSR_rpt_1']"
```

CognosPackageUpdate

The `CognosPackageUpdate` verb modifies a package or report data source definition in the model. The package or report definition is a reference to a package or report that is defined in IBM Cognos 8. You can then create a data source definition which has columns based on items that are selected from the package or report.

Notes:

- If you include a data source ID for the new package or report in the syntax, and the object doesn't exist, you will receive an error message.
- To insert a dimension from a package (the **Insert Dimension From Package** command on the **Edit** menu, you must first add the package and data source and then add a dimension from the data source. For more information about adding a dimension from a data source, see "DimAdd" (p. 145).

The syntax for a query based on an IBM Cognos 8 package or report is as follows:

CognosPackageUpdate *objDataSource* [cognospackage_opts]

Argument	Description
CognosPackageUpdate *objDataSource*	Creates the data source *objDataSource*. *objDataSource* must be the object name and can include the object identifier.
cognospackage_opts	Optional parameters that can describe the package or report in greater detail. For the complete list of options, see "cognospackageopts" (p. 218).

Example

This example modifies the IBM Cognos 8 package data source Go Sales and Retailers.

```
CognosPackageUpdate 103 "GO Sales and Retailers" SourceType
Package SourcePath "/content/package[@name='GO Sales
and Retailers']" PackageTimeStamp "/content/package[@name='GO Sales
and Retailers']/model[@name='model']"
```

This example modifies the IBM Cognos 8 report data base GSR_rpt_1.

```
CognosPackageUpdate 115 "GSR_rpt_1" SourceType Report SourcePath
"/content/package[@name='GO Salesand Retailers']/report[@name='GSR_rpt_1']"
PackageTimeStamp"/content/package[@name='GO
Sales and Retailers']/report[@name='GSR_rpt_1']"
```

CognosPackageDatasourceConnectionAdd

The `CognosPackageDatasourceConnectionAdd` verb defines the data source signon to be used when the data source has an ambiguous signon: multiple connections, each with multiple signons.

The equivalent on the Windows interface is to define a Transformer data source signon when prompted.

The syntax is as follows:

```
CognosPackageDatasourceConnectionAdd "Content Store data source name"
CognosPackageConnection "connection" CognosPackageConnectionSignon "ContentStore
signon name" Database Transformer data source ID "Transformerdata
source signon name" CognosPackageAlwaysUseTransformerSignon {True|False}
CognosPackagePowercubeSource {True|False}
```

Argument	Description
CognosPackageDatasourceConnection-Add "*Content Store data source name*"	Specifies that a Transformer signon is added for the IBM Cognos 8 data source defined in Content Manager.
CognosPackageConnection "*connection*"	Specifies the connection to the IBM Cognos 8 data source defined in Content Manager.
CognosPackageConnectionSignon "*Content Store signon name*"	Specifies the signon name defined in Content Manager.
Database *Transformer data source ID* "*Transformer data source signon name*"	Specifies the Transformer data source ID and new Transformer data source signon name.
CognosPackageAlwaysUseTransformer-Signon {True|False}	Specifies whether queries use the new Transformer signon or the signon defined in Content Manager.
CognosPackagePowercubeSource {True|False}	Specifies whether the data source is a PowerCube.

Example

This example adds the Transformer signon "cogtr_signon" for the oracle_gosales data source with data source connection "connection1" and signon "signon1" defined in Content Manager. Queries will always use the "cogtr_signon" signon. The data source is not an IBM Cognos PowerCube.

```
CognosPackageDatasourceConnectionAdd "oracle_gosales"
```

```
CognosPackageConnection "connection1"
CognosPackageConnectionSignon "signon1"
Database 181 "cogtr_signon"
CognosPackageAlwaysUseTransformerSignon True CognosPackagePowercubeSource False
```

CognosPackageDatasourceConnectionDelete

The `CognosPackageDatasourceConnectionDelete` verb deletes a previously-created Transformer data source signon.

The equivalent on the Windows interface is to click **Delete Signon** on the edit menu when a signon is selected.

The syntax is as follows:

CognosPackageDatasourceConnectionDelete *"Content Store data source name"*

Argument	Description
CognosPackageDatasourceConnectionDelete "*Content Store data source name*"	Specifies that a Transformer signon is deleted for the IBM Cognos 8 data source defined in Content Manager.

Example

This example deletes the Transformer signon "cogtr_signon".

```
CognosPackageDatasourceConnectionDelete "oracle_gosales"
```

CognosPackageDatasourceConnectionMake

The `CognosPackageDatasourceConnectionMake` verb creates the data source signon to be used when the data source has an ambiguous signon: multiple connections, each with multiple signons.

The equivalent on the Windows interface is to define a Transformer data source signon when prompted.

The syntax is as follows:

CognosPackageDatasourceConnectionMake *"Content Store data source name"* **CognosPackageConnection** *"connection"* **CognosPackageConnectionSignon** *"ContentStore signon name"* **Database** *Transformer data source ID "Transformerdata source signon name"* **CognosPackageAlwaysUseTransformerSignon** {True|False} **CognosPackagePowercubeSource** {True|False}

Argument	Description
CognosPackageDatasourceConnectionMake "*Content Store data source name*"	Specifies that a Transformer signon is created for the IBM Cognos 8 data source defined in Content Manager.

Argument	Description
CognosPackageConnection "*connection*"	Specifies the connection to the IBM Cognos 8 data source defined in Content Manager.
CognosPackageConnectionSignon "*Content Store signon name*"	Specifies the signon name defined in Content Manager.
Database *Transformer data source ID* "*Transformer data source signon name*"	Specifies the Transformer data source ID and new Transformer data source signon name.
CognosPackageAlwaysUseTransformerSignon {True|False}	Specifies whether queries use the new Transformer signon or the signon defined in Content Manager.
CognosPackagePowercubeSource {True|False}	Specifies whether the data source is a PowerCube.

Example

This example creates the Transformer signon "cogtr_signon" for the oracle_gosales data source with data source connection "connection1" and signon "signon1" defined in Content Manager. Queries will always use the "cogtr_signon" signon. The data source is not an IBM Cognos PowerCube.

```
CognosPackageDatasourceConnectionMake "oracle_gosales"
CognosPackageConnection "connection1"
CognosPackageConnectionSignon "signon1" Database 181
"cogtr_signon"
CognosPackageAlwaysUseTransformerSignon True
CognosPackagePowercubeSource False
```

CognosPackageDatasourceConnectionUpdate

The `CognosPackageDatasourceConnectionUpdate` verb updates the data source signon created when the data source has an ambiguous signon: multiple connections, each with multiple signons.

The equivalent on the Windows interface is to modify the Transformer data source signon property sheet.

The syntax is as follows:

CognosPackageDatasourceConnectionUpdate *"Content Store data source name"* **CognosPackageConnection** *"connection"* **CognosPackageConnectionSignon** *"ContentStore signon name"* **Database** *Transformer data source ID "Transformerdata source signon name"* **CognosPackageAlwaysUseTransformerSignon** {True|False} **CognosPackagePowercubeSource** {True|False}

Argument	Description
CognosPackageDatasourceConnectionUpdate "*Content Store data source name*"	Specifies that a Transformer signon is updated for the IBM Cognos 8 data source defined in Content Manager.
CognosPackageConnection "*connection*"	Specifies the connection to the IBM Cognos 8 data source defined in Content Manager.
CognosPackageConnectionSignon "*Content Store signon name*"	Specifies the signon name defined in Content Manager.
Database *Transformer data source ID* "*Transformer data source signon name*"	Specifies the Transformer data source ID and Transformer data source signon name.
CognosPackageAlwaysUseTransformerSignon {True|False}	Specifies whether queries use the new Transformer signon or the signon defined in Content Manager.
CognosPackagePowercubeSource {True|False}	Specifies whether the data source is a PowerCube.

Example

This example updates the Transformer signon "cogtr_signon" for the oracle_gosales data source with data source connection "connection1" and signon "signon1" defined in Content Manager. Queries will no longer always use the "cogtr_signon" signon. The data source is not an IBM Cognos PowerCube.

```
CognosPackageDatasourceConnectionUpdate "oracle_gosales"
CognosPackageConnection "connection1"
CognosPackageConnectionSignon "signon1"
Database 181 "cogtr_signon"
CognosPackageAlwaysUseTransformerSignon False
CognosPackagePowercubeSource False
```

CognosPackageFilterAdd

The `CognosPackageFilterAdd` verb imports filters defined within IBM Cognos 8 package or report data sources.

The syntax is as follows:

```
CognosPackageFilterAdd "filter name" DATASOURCE data source ID
CognosPackageFilterRef "filter reference ID"
```

Argument	Description
CognosPackageFilterAdd "*filter name*"	Specifies the name of the filter to be added in the Transformer model.
DATASOURCE *data source ID*	Specifies the IBM Cognos 8 data source ID.
CognosPackageFilterRef "*filter reference ID*"	Specifies the reference ID of the filter to be added in the Transformer model.

Example

The following example adds the package filter "Tents1" in the Transformer model, using the IBM Cognos 8 filter reference `oracle_gosales.Tents` from the IBM Cognos 8 data source with ID 105.

```
CognosPackageFilterAdd "Tents1" DATASOURCE 105 CognosPackageFilterRef "[oracle_
gosales].[Tents]"
```

CognosPackageFilterDelete

The `CognosPackageFilterDelete` verb deletes filters imported with IBM Cognos 8 package or report data sources.

The syntax is as follows:

```
CognosPackageFilterDelete "filter name"
```

Argument	Description
CognosPackageFilterDelete "*filter name*"	Specifies the name of the filter to be deleted from the Transformer model.

Example

The following example deletes the filter "Tents1" from the Transformer model.

```
CognosPackageFilterDelete "Tents1"
```

CognosPackageFilterMake

The `CognosPackageFilterMake`verb imports filters defined within IBM Cognos 8 package or report data sources.

The syntax is as follows:

```
CognosPackageFilterMake "filter name" DATASOURCE data source ID
CognosPackageFilterRef "filter reference ID"
```

Argument	Description
CognosPackageFilterMake "*filter name*"	Specifies the name of the filter to be added in the Transformer model.
DATASOURCE *data source ID*	Specifies the IBM Cognos 8 data source ID.
CognosPackageFilterRef "*filter reference ID*"	Specifies the reference ID of the filter to be added in the Transformer model.

Example

The following example creates the package filter "Tents1" in the Transformer model, using the IBM Cognos 8 filter reference `oracle_gosales.Tents` from the IBM Cognos 8 data source with ID 105.

```
CognosPackageFilterMake "Tents1" DATASOURCE 105 CognosPackageFilterRef "[oracle_
gosales].[Tents]"
```

CognosPackageFilterUpdate

The `CognosPackageFilterUpdate` verb updates filters imported with IBM Cognos 8 package or report data sources.

The syntax is as follows:

```
CognosPackageFilterUpdate "filter name" DATASOURCE data source ID
CognosPackageFilterRef "filter reference ID"
```

Argument	Description
CognosPackageFilterUpdate "*filter name*"	Specifies the name of the filter to be updated in the Transformer model.
DATASOURCE *data source ID*	Specifies the IBM Cognos 8 data source ID.
CognosPackageFilterRef "*filter reference ID*"	Specifies the reference ID of the filter to be added in the Transformer model.

Example

The following example updates the filter "Tents1" to use the package filter "[oracle_gosales].[Tents2]".

```
CognosPackageFilterUpdate "Tents1" DATASOURCE 105 CognosPackageFilterRef "
[oracle_gosales].[Tents2]"
```

ColumnAdd

The `ColumnAdd` verb adds a column to a data source.

Its Windows interface equivalent is the **Insert Column** command on the **Edit** menu.

For more information about creating columns, see "ColumnMake" (p. 111).

The syntax is as follows:

```
ColumnAdd objCol [
DataSource objDataSource]
Origin origin Offset offset [
colopts]
```

Argument	Description
ColumnAdd *objCol*	Creates the column *objCol*. *objCol* must be the object name, and can include the object identifier.
DataSource *objDataSource*	Places the column within a data source. *objDataSource* can be the data source object name, object identifier, or both.
Origin *origin*	Specifies the origin. *origin* must be one of `Source`, `Generated`, `Calculated`, or `Manual`. This argument is mandatory.
Offset *offset*	Specifies the amount by which the column is offset. *offset* can be `0` or greater. This argument is mandatory.
colopts	Parameters that describe the column in greater detail. All are optional except `Column`*objCol*, which is mandatory. Some options are set by default if you do not set them. For the complete list of options, see colopts.

Example

This example adds the Branch Code column to the model.

```
ColumnAdd "Branch Code" DataSource "All Staff Count (Excel)" Origin Source
Offset 1 Column "Branch Code"
```

ColumnDelete

The `ColumnDelete` verb removes a column.

The Windows interface equivalent is to click **Delete** on the **Edit** menu when a column is selected.

The syntax is as follows:

```
ColumnDelete objCol [
DataSource objDataSource]
```

Argument	Description
ColumnDelete *objCol*	Specifies the column that is to be deleted. *objCol* must be the object name, and can include its object identifier.
DataSource *objDataSource*	Is specified, if necessary, to uniquely identify the column. *objDim* can be the dimension object name, object identifier, or both.

Example

This example deletes the column Branch Code.

```
ColumnDelete "Branch Code" DataSource "Locations (CSV)"
```

ColumnListUpdate

The `ColumnListUpdate` verb reorders the columns in a data source.

The equivalent operation on the Windows interface is to drag and drop the columns as required.

The syntax is as follows:

ColumnListUpdate [**DataSource** *objDataSource*] **StartList** *objCols* **EndList**

Argument	Description
DataSource *objDataSource*	Specifies the data source to be reordered. *objDataSource* can be the data source object name, object identifier, or both.
StartList *objCols*	Lists the columns in the desired order. If not all columns are listed, the listed ones move to the top of the list. The columns can be identified by object name or object identifier.

Example

This example reorders the columns in the Excel data source "All Staff Count" so that **Branch Code** is first, followed by **Time** and **Staff Count**.

```
ColumnListUpdate DataSource "All Staff Count (Excel)" StartList "Branch Code"
"Time" "Staff Count" EndList
```

ColumnMake

The `ColumnMake` verb creates a column or updates an existing one.

The Windows interface equivalent, if the column exists, is to modify its **Column** property sheet. For a new column, click **Insert Column** on the **Edit** menu.

Column names cannot contain an at sign (`@`).

The syntax is as follows:

```
ColumnMake objCol [
DataSource objDataSource]
Origin origin Offset offset [
colopts]
```

Argument	Description
ColumnMake *objCol*	Creates the column *objCol* or modifies it if it exists. *objCol* can be the column object name, object identifier, or both. Include the object name if the column does not exist.
DataSource *objDataSource*	When creating a column, this specifies the data source and where it is to be placed. When updating a column this is used, if necessary, to uniquely identify it. *objDataSource* can be the object name, object identifier, or both.
Origin *origin*	Specifies the origin. *origin* must be one of `Source`, `Generated`, `Calculated`, or `Manual`. This argument is mandatory.
Offset *offset*	Specifies the amount by which the column is offset. *offset* can be equal or greater than `0`. This argument is mandatory.
colopts	Parameters that describe the column in greater detail. All are optional except `Column`*objCol*, which is mandatory. Some options are set by default if you do not set them. For the complete list of options, see colopts. If the column exists, previously set options are retained unless you change them with this command.

Example

This example defines the column **Branch Code**.

```
ColumnMake "Branch Code" DataSource "All Staff Count (Excel)" Origin Source
Offset 1 Column "Branch Code" Storage Default Scale 0 Size 1 Decimals 0
InputScale 0 TimeArray Off
```

ColumnUpdate

The `ColumnUpdate` verb updates an existing column.

The equivalent action on the Windows interface is to modify the **Column** property sheet.

For more information about updating columns, see "ColumnMake" (p. 111).

The syntax is as follows:

```
ColumnUpdate objCol [
DataSource objDataSource]
[colopts]
```

Argument	Description
ColumnUpdate *objCol*	Specifies the column to update. *objCol* can be the object name, object identifier, or both.
DataSource *objDataSource*	Is specified, if necessary, to uniquely identify the column. *objDataSource* can be the object name, object identifier, or both.
colopts	Optional parameters that describe the column in greater detail. For the complete list of options, see colopts. Previously set options are retained unless you change them with this command.

Example

This example changes the date format of the Time column to the Year-Month form, and changes the degree level for the date to Month.

```
ColumnUpdate "Time" DataSource "Salesrep Plan (Excel)" Format YM DateLevel
Month
```

CreateColumns

The `CreateColumns` verb creates columns for data sources. There is no Windows interface equivalent because columns are created automatically in the **Data Sources** list on the Windows interface.

The `CreateColumns` verb does not work when the associated data source contains the setting `Columns True`.

The syntax is as follows:

```
CreateColumns objQueries
```

Argument	Description
CreateColumns *objQueries*	*objQueries* are one or more data sources for which columns are to be created. The list must be made up of object names or object identifiers.

Example

This example creates columns for the data source **Products (CSV)**.

```
CreateColumns "Products (CSV)"
```

CreateFiles

The `CreateFiles` verb creates an .mdc file for each PowerCube in the model.

The Windows interface equivalent is the **Create PowerCube** toolbar button or the **Create PowerCubes** command on the **Run** menu.

For other ways to create and update .mdc files, see "CreateFromCubes" (p. 114), "CreateFromQueries" (p. 115), and "UpdatePowerCubes" (p. 191).

The syntax is as follows:

```
CreateFiles
```

Example

This example creates .mdc files for every cube in the model.

```
CreateFiles
```

CreateFromCubes

The `CreateFromCubes` verb creates an .mdc file for each specified PowerCube.

The Windows interface equivalent is to click **Create Selected PowerCubes** on the **Run** menu when one or more PowerCubes are selected.

For other ways to create and update .mdc files, see "CreateFiles" (p. 114), "CreateFromQueries" (p. 115), and "UpdatePowerCubes" (p. 191).

The syntax is as follows:

```
CreateFromCubes [objCubes]
```

Argument	Description
objCubes	Specifies one or more PowerCubes for which .mdc files are to be created. PowerCubes can be identified by object name or object identifier.

Example

This example creates an .mdc file for the PowerCube **cube1**.

```
CreateFromCubes "cube1"
```

CreateFromQueries

The `CreateFromQueries` verb creates an .mdc file for each PowerCube in the model by retrieving data from only specific data sources. There is no Windows interface equivalent.

If no data sources are specified, nothing happens.

If one or more of the specified data sources are not set for **PowerCube Creation** on the **General** tab of the **Data Source** property sheet in the Windows interface, an error is issued. This can occur with structural data sources because they do not contain transactional data.

For other ways to create and update .mdc files, see "CreateFromCubes" (p. 114), "CreateFiles" (p. 114), and "UpdatePowerCubes" (p. 191).

The syntax is as follows:

`CreateFromQueries` `[`*`objQueries`*`]`

Argument	Description
objQueries	Specifies one or more data sources that are used to create .mdc files for the PowerCubes. Data sources can be identified by object name or object identifier.

Example

This example creates .mdc files from the **All Staff Count (Excel)** data source.

```
CreateFromQueries "All Staff Count(Excel)"
```

CubeAdd

The `CubeAdd` verb creates the PowerCube object in the Transformer model. To create or update an .mdc file, see "CreateFiles" (p. 114).

The Windows interface equivalent is to select the **Insert** option on the **Edit** menu when the **PowerCubes** pane is selected.

For more information about creating PowerCubes, see "CubeMake" (p. 125).

The syntax is as follows:

`CubeAdd` *`objCube`* `[`
`powercubeopts``]` `[`**`DimensionView`** *`objView`*`][`**`MeasureInclude`** *`objMeasure meaopt`*`]`

Argument	Description
CubeAdd *objCube*	Creates the PowerCube *objCube*. *objCube* must be the object name and can include the object identifier.

Argument	Description
powercubeopts	Optional parameters that describe the cube in greater detail. Some options are set by default if you do not set them. For the complete list of options, see powercubeopts.
DimensionView *objDim objView*	Specifies the dimension and dimension view. *objDim* can be the object name or object identifier. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in `DimensionView "Years" "All Categories" DimensionView "Products" "All Categories"`
MeasureInclude *objMeasure meaopt*	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. *meaopt* must be one of `Yes`, `Indirect`, or `No`. Repeat this argument to specify multiple measures, as in `MeasureInclude 159 Yes MeasureInclude 175 No`

Example

This example creates the **Great Outdoors Company Sales** PowerCube object.

```
CubeAdd "Great Outdoors Company Sales" MdcFile "c:\outdoors.mdc"
```

CubeDelete

The `CubeDelete` verb removes the PowerCube object from the Transformer model.

The Windows interface equivalent is to click **Delete** on the **Edit** menu when a cube is selected.

The `CubeDelete` verb only deletes the PowerCube object, not the .mdc file.

The syntax is as follows:

`CubeDelete` *`objCube`*

Argument	Description
CubeDelete *objCube*	Specifies the PowerCube to be deleted. *objCube* can be the object name, object identifier, or both.

Example

This example deletes the PowerCube called Great Outdoors Company Sales.

```
CubeDelete "Great Outdoors Company Sales"
```

CubeGroupAdd

The `CubeGroupAdd` verb creates the `PowerCubeGroup` object in the Transformer model. To create or update an .mdc file, see "CreateFiles" (p. 114).

The Windows interface equivalent is to select the **Insert** option from the **Edit** menu when the **PowerCubes** pane is selected, and then to specify a dimension and level on the **Cube Group** tab.

If you do not include the options `SegmenterLevel` and `SegmenterDimension`, the `CubeGroupAdd` verb creates a PowerCube rather than a `PowerCubeGroupCube` object. For more information, see powercubeopts.

For more information about updating cube groups, see "CubeGroupMake" (p. 122).

The syntax is as follows:

CubeGroupAdd *objCubeGroup* [
powercubeopts] [**DimensionView** *objDim objView*] [**MeasureInclude** *objMeasure meaopt*]

Argument	Description
CubeGroupAdd *objCubeGroup*	Specifies the cube group that is to be created. *objCubeGroup* must be the object name, and can also be the object identifier.
powercubeopts	Optional parameters that describe the cube in greater detail. Some options are set by default if you do not set them. For the complete list of options, see powercubeopts.
DimensionView *objDim objView*	Specifies the dimension and dimension view. *objDim* can be the object name or object identifier. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in `DimensionView "Years" "All Categories" DimensionView "Products" "All Categories"`
MeasureInclude *objMeasure meaopt*	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. *meaopt* must be one of `Yes`, `Indirect` or `No`. Repeat this argument to specify multiple measures, as in `MeasureInclude 159 Yes MeasureInclude 175 No`

Example

This example creates the cube group cubegroup1. Cubes within this group are created from the Products dimension and the Product Line level.

```
CubeGroupAdd "cubegroup1" MdcFile "c:\temp\outdoors1.mdc" SegmenterDimension
"Products" SegmenterLevel "Product Line"
```

CubeGroupCubeAdd

The `CubeGroupCubeAdd` verb creates the PowerCube object for a `PowerCubeGroup` object in the Transformer model. To create or update an .mdc file, see "CreateFiles" (p. 114).

The Windows interface equivalent is to select the **Insert** option from the **Edit** menu when the **PowerCubes** pane is selected, and then to provide a dimension and level on the **Cube Group** tab.

In the Windows interface, a `PowerCubeGroupCube` object is created for every category under the Drill category in the dimension, unless some categories have been excluded, cloaked, or suppressed. In MDL, a separate statement must be included for every category under the Drill category.

For more information about creating cubes in cube groups, see "CubeGroupCubeMake" (p. 120).

The syntax is as follows:

```
CubeGroupCubeAdd objCubeGroupCubeCubeGroup objCubeGroup [
Dimension objDim]
Code objCat [
powercubeopts]
```

Argument	Description
CubeGroupCubeAdd *objCubeGroup*	Creates the PowerCube group cube *objCubeGroupCube*. *objCubeGroupCube* must be the object name and can include the object identifier.
CubeGroup *objCubeGroup*	Specifies a cube group for the PowerCube. *objCubeGroup* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the category. *objDim* can be the object name, object identifier, or both.
Code *objCat*	Specifies the category on which the PowerCube is based. *objCat* can be the object name, object identifier, or both. This argument is mandatory.

Argument	Description
powercubeopts	Optional parameters that describe the cube in greater detail. Some options are set by default if you do not set them. For the complete list of options, see powercubeopts.

Example

This example creates the **Outdoor Products** cube within the cube group **cubegroup1**, based on the **Outdoor Products** category.

```
CubeGroupCubeAdd "Outdoor Products" CubeGroup "cubegroup1" Code "Outdoor
Products"
```

CubeGroupCubeDelete

The `CubeGroupCubeDelete` verb removes the PowerCube from a `PowerCubeGroupCube` object in the Transformer model. It does not delete the .mdc file.

The Windows interface equivalent is to click **Delete** on the **Edit** menu when a PowerCube in a cube group is selected.

This verb can only be used if the model contains a dimension view that apexes, cloaks, suppresses, or excludes the category at the apex of the cube.

A `PowerCubeGroupCube` object can only be deleted if it is suppressed in the dimension view that underlies the cube group. This means that you must first create a dimension view that apexes, cloaks, suppresses, or excludes the category that defines the `PowerCubeGroupCube` object.

The syntax is as follows:

CubeGroupCubeDelete *objCubeGroupCube* [
CubeGroup *objCubeGroup*]

Argument	Description
CubeGroupCubeDelete *objCubeGroupCube*	Deletes the cube group *objCubeGroupCube*. *objCubeGroupCube* can be the object name, object identifier, or both.
CubeGroup *objCubeGroup*	Specifies the cube group in which the PowerCube is located. *objCubeGroup* can be the object name, object identifier, or both.

Example

This example deletes the Environmental Line cube from cubegroup1. Before the deletion, a dimension view is created to suppress the category.

```
ViewMake "Environmental Line" Dimension "Products" ViewSecurity 0 Apex
"Product Line" Suppressed "Environmental Line"
CubeGroupCubeDelete "Environmental Line" CubeGroup "cubegroup1"
```

CubeGroupCubeListUpdate

The `CubeGroupCubeListUpdate` verb reorders the PowerCubes in a `PowerCubeGroup` object.

The Windows interface equivalent is to drag PowerCubes into the appropriate cube group.

The syntax is as follows:

CubeGroupCubeListUpdateCubeGroup *objCubeGroup* [**StartList** *objCubeGroups* **EndList**]

Argument	Description
CubeGroupCubeListUpdate CubeGroup *objCubeGroup*	Specifies the cube group. *objCubeGroup* can be the object name, object identifier, or both.
StartList *objCubeGroupCubes*	Specifies the cube group cubes in the desired order. Each cube can be identified by object name or object identifier. If no cubes are specified, nothing happens. If some of the cubes are specified, they move to the top of the list.

Example

This example places the cubes GO Sport Line and Outdoor Products at the top of the list of cubes in cubegroup1.

```
CubeGroupCubeListUpdate CubeGroup "cubegroup1" StartList "GO Sport Line"
"Outdoor Products" EndList
```

CubeGroupCubeMake

The `CubeGroupCubeMake` verb creates a PowerCube in a `PowerCubeGroup` object, or updates it if it exists. It does so by defining a PowerCube group cube object in the Transformer model. To create or update an .mdc file, see "CreateFiles" (p. 114).

The Windows interface equivalent, if the cube already exists within a cube group, is to modify the **Cube** property sheet. For a new cube in a cube group, select the **Insert** option on the **Edit** menu when the PowerCubes pane is selected, and then provide a dimension and level on the **Cube Group** tab.

The cube group must exist before you can create a PowerCube for the group.

Note: Category object names differ between MDL and the Windows interface. MDL uses the category code as the object name. For more information, see "Locating Objects Uniquely".

The syntax is as follows:

```
CubeGroupCubeMake objCubeGroupCube [
CubeGroup objCubeGroup]
[Dimension objDim]
Code objCat [
powercubeopts]
```

Argument	Description
CubeGroupCubeMake *objCubeGroupCube*	Creates the PowerCube group cube *objCubeGroupCube* or modifies it if it exists. *objCubeGroupCube* can be the object name, object identifier, or both. Include the object name if the cube group cube does not exist.
CubeGroup *objCubeGroup*	When creating a cube in a cube group, this specifies the cube group. *objCubeGroup* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the category. *objDim* can be the object name, object identifier, or both.
Code *objCat*	Specifies the category on which the cube is based. *objCat* can be the object name, object identifier, or both.
powercubeopts	Optional parameters that describe the cube in greater detail. Some options are set by default if you do not set them. For the complete list of options, see powercubeopts.

Example

This example defines the PowerCube group cube Outdoor Products.

```
CubeGroupCubeMake 6343 "Outdoor Products"  CubeGroup "cubegroup1" Code 4807
Status OK  CubeCreation Default Optimize Default Compress
False IncrementalUpdate False ServerCube False  CubeStamp 886518687 CubeCycle
24 DrillThrough False EndList
```

CubeGroupCubeUpdate

The `CubeGroupCubeUpdate` verb updates an existing PowerCube within a `PowerCubeGroup` object in the Transformer model. It does not modify existing .mdc files.

The Windows interface equivalent is to modify the **Cube Group** property sheet.

For more information about updating cubes in cube groups, see "CubeGroupCubeMake" (p. 120).

The syntax is as follows:

```
CubeGroupCubeUpdate objCubeGroupCube [
CubeGroup objCubeGroup] [
powercubeopts]
```

Argument	Description
CubeGroupCubeUpdate *objCubeGroupCube*	Specifies the cube group cube to update. *objCubeGroupCube* can be the object name, object identifier, or both.
CubeGroup *objCubeGroup*	Specifies the cube group. *objCubeGroup* can be the object name, object identifier, or both.
powercubeopts	Optional parameters that describe the cube in greater detail. For the complete list of options, see powercubeopts. Previously set options are retained unless you change them with this command.

Example

This example modifies the cube Outdoor Products from cubegroup1 by enabling drill-through.

```
CubeGroupCubeUpdate "Outdoor Products" CubeGroup "cubegroup1"
DrillThrough True EndList
```

CubeGroupDelete

The `CubeGroupDelete` verb removes a `PowerCubeGroup` object from the Transformer model. It does not delete the .mdc file.

The Windows interface equivalent is to click **Delete** on the **Edit** menu when a cube group is selected.

The syntax is as follows:

`CubeGroupDelete` *`objCubeGroup`*

Argument	Description
CubeGroupDelete *objCubeGroup*	Specifies the cube group that is to be deleted. *objCubeGroup* can be the object name, object identifier, or both.

Example

This example deletes the PowerCube cubegroup1.

```
CubeGroupDelete "cubegroup1"
```

CubeGroupMake

The `CubeGroupMake` verb creates a `PowerCubeGroup` object or updates an existing one, by defining the PowerCube object in the Transformer model. To create or update an .mdc file, see "Create-Files" (p. 114).

The Windows interface equivalent, for an existing cube group, is to modify its **Cube Group** property sheet. For a new cube group, select the **Insert** option from the **Edit** menu when the **PowerCubes** pane is selected, and then provide a dimension and level on the **Cube Group** tab.

If you do not include the options `SegmenterLevel` and `SegmenterDimension`, the `CubeGroupMake` verb will create a PowerCube rather than a `PowerCubeGroupCube` object. For more information, see powercubeopts.

The syntax is as follows:

```
CubeGroupMake objCubeGroup [
powercubeopts] [DimensionView objDim objView]
[MeasureInclude objMeasure meaopt]
```

Argument	Description
CubeGroupMake *objCubeGroup*	Creates the cube group *objCubeGroup* or modifies it if it exists. *objCubeGroup* can be the object name, object identifier, or both. Include the object name if the category does not exist.
powercubeopts	Optional parameters that describe the cube in greater detail. Some options are set by default if you do not set them. For the complete list of options, see powercubeopts. If the cube exists, previously set options are retained unless you change them with this command.
DimensionView *objDim objView*	Specifies the dimension and dimension view. *objDim* can be the object name or object identifier. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in `DimensionView "Years" "All Categories" DimensionView "Products" "All Categories"`
MeasureInclude *objMeasure meaopt*	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. *meaopt* must be one of `Yes`, `Indirect`, or `No`. Repeat this argument to specify multiple measures, as in `MeasureInclude "Cost" Yes MeasureInclude "Revenue" No`

Example

This example defines the cube group cubegroup1. The cubes within this group are created from the Products dimension, at the Product Line level with a focus of Product Type.

```
CubeGroupMake "cubegroup1" MdcFile "c:\outdoors1.mdc" CubeCreation On
Optimize Default Compress False  DatabaseInfo "Local;; "IncrementalUpdate
False ServerCube False DrillThrough False EndList SegmenterDimension
"Products" SegmenterLevel "Product Line" DetailLevel "Product
Type" DimensionView "Years" "All Categories" DimensionView "Products" "All
Categories"  DimensionView "Locations" "All Categories" DimensionView
"Channels" "All Categories" DimensionView "Margin Ranges" "All
Categories" MeasureInclude "Revenue" Yes MeasureInclude "Product Cost"
Yes MeasureInclude "Product Plan" Yes MeasureInclude "Expense Plan"
Yes MeasureInclude "Quantity Sold" Yes MeasureInclude "Profit Margin %"
Yes MeasureInclude "Revenue/Employee" Yes MeasureInclude "SalesRep Plan"
Yes MeasureInclude "Staff Count" Yes MeasureInclude "SalesRep Count" Yes
```

CubeGroupUpdate

The `CubeGroupUpdate` verb updates an existing `PowerCubeGroup` object by modifying the PowerCube in the Transformer model. It does not affect existing .mdc files.

The Windows interface equivalent is to modify the **Cube Group** property sheet.

For more information about updating cube groups, see "CubeGroupMake" (p. 122).

The syntax is as follows:

CubeGroupUpdate *objCubeGroup* [
powercubeopts] [**DimensionView** *objDim objView*]
[**MeasureInclude** *objMeasure meaopt*]

Argument	Description
CubeGroupUpdate *objCubeGroup*	Specifies the PowerCube group to update. *objCubeGroup* can be the object name, object identifier, or both.
powercubeopts	Optional parameters that describe the cube in greater detail. For the complete list of options, see powercubeopts. Previously set options are retained unless you change them with this command.
DimensionView *objDim objView*	Specifies the dimension and dimension view. *objDim* can be the object name or object identifier. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in `DimensionView "Years" "All Categories" DimensionView "Products" "All Categories"`

Argument	Description
MeasureInclude *objMeasure meaopt*	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. *meaopt* must be one of `Yes`, `Indirect`, or `No`. Repeat this argument to specify multiple measures, as in `MeasureInclude 159 Yes MeasureInclude 175 No`

Example

This example modifies Cubegroup1 by changing the dimension view from Locations to Omit Dimension and by excluding the measure Product Cost from the view.

```
CubeGroupUpdate "Cubegroup1" DimensionView "Locations" "Omit Dimension"
MeasureInclude "Product Cost" Indirect
```

CubeMake

The `CubeMake` verb creates a PowerCube or updates an existing one by defining the PowerCube object in the Transformer model. To create or update an .mdc file, see "CreateFiles" (p. 114).

The Windows interface equivalent, if the cube exists, is to modify the **Cube** property sheet. For a new cube, when the **PowerCubes** pane is selected, select the **Insert** option on the **Edit** menu.

The syntax is as follows:

```
CubeMake objCube [
powercubeopts] [DimensionView objDim objView][
MeasureInclude objMeasuremeaopt]
```

Argument	Description
CubeMake *objCube*	Creates the PowerCube *objCube* or modifies it if it exists. *objCube* can be the object name, object identifier, or both. Include the object name if the category does not exist.
powercubeopts	Optional parameters that describe the cube in greater detail. Some options are set by default if you do not set them. For the complete list of options, see powercubeopts. If the cube exists, previously set options are retained unless you change them with this command.

Argument	Description
DimensionView *objDim objView*	Specifies the dimension and dimension view. *objDim* can be the object name or object identifier. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in `DimensionView "Years" "All Categories" DimensionView "Products" "All Categories"`
MeasureInclude *objMeasure meaopt*	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. *meaopt* must be one of `Yes`, `Indirect`, or `No`. Repeat this argument to specify multiple measures, as in `MeasureInclude 159 Yes MeasureInclude 175 No`

Example

This example defines the PowerCube Great Outdoors Company Sales.

```
CubeMake "Great Outdoors Company Sales"  MdcFile "c:\outdoors.mdc" Status OK
CubeCreation On Optimize Categories Compress False DatabaseInfo "Local;;
" IncrementalUpdate False ServerCube False CubeStamp 886096730
CubeCycle 6 DrillThrough False EndList DimensionView "Years" "All
Categories" DimensionView "Products" "All Categories" DimensionView
"Locations" "All Categories" DimensionView "Channels" "All
Categories" DimensionView "Margin Ranges" "All Categories" MeasureInclude
"Revenue" Yes MeasureInclude "Product Cost" Yes MeasureInclude "Product Plan"
Yes MeasureInclude "Expense Plan" Yes MeasureInclude "Quantity Sold"
Yes MeasureInclude "Profit Margin %" Yes MeasureInclude "Revenue/Employee"
Yes MeasureInclude "SalesRep Plan" Yes MeasureInclude "Staff Count"
Yes MeasureInclude "SalesRep Count" Yes
```

CubeUpdate

The `CubeUpdate` verb updates an existing PowerCube by modifying the PowerCube object in the Transformer model. To create or update an .mdc file, see "CreateFiles" (p. 114).

The Windows interface equivalent is to modify the **PowerCube** property sheet.

For more information about updating PowerCubes, see "CubeMake" (p. 125).

The syntax is as follows:

CubeUpdate *objCube* [
powercubeopts] [**DimensionView** *objDim objView*][
MeasureInclude *objMeasure meaopt*]

Argument	Description
CubeUpdate *objCube*	Specifies the PowerCube to update. *objCube* can be the object name, object identifier, or both.
powercubeopts	Optional parameters that describe the cube in greater detail. For the complete list of options, see powercubeopts. If the cube exists, previously set options are retained unless you change them with this command.
DimensionView *objDim objView*	Specifies the dimension and dimension view. *objDim* can be the object name or object identifier. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views, as in `DimensionView "Years" "All Categories" DimensionView "Products" "All Categories"`
MeasureInclude *objMeasure meaopt*	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. *meaopt* must be one of `Yes`, `Indirect`, or `No`. Repeat this argument to specify multiple measures, as in `MeasureInclude 159 Yes MeasureInclude 175 No`

Example

This example modifies the PowerCube Great Outdoors Company Sales by enabling drill-through, changing the dimension view to Omit Dimension and excluding the Revenue measure.

```
CubeUpdate "Great Outdoors Company Sales" MdcFile "c:\outdoors.mdc"
DrillThrough True EndList DimensionView "Products" "Omit
Dimension" MeasureInclude "Revenue" Indirect
```

CurrencyAdd

The `CurrencyAdd` verb adds a currency definition to the currency table for the model.

The Windows interface equivalent is to right-click a currency item in the currency table and then select **Add New Currency**.

The syntax is as follows.

```
CurrencyAdd objCurrency [
Dimension objDim]
[Drill objDrillCat]
Levels objLevel
[currencyrecordopts] CurrencyRateListEffectiveDate {
```

```
objCat|0} ConversionRate rate [RateIsBySource {True|False}]
[CurrencyTableType tabletype]
EndList
```

Argument	Description
CurrencyAdd *objCurrency*	Adds the currency *objCurrency* to the list of currencies in the currency table. *objCurrency* must be the object name and can include the object identifier.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the level *objLevel*. *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Is specified, if necessary, to uniquely identify the date level *objLevel*. *objDrillCat* can be the drill category object name, object identifier, or both.
Levels *objLevel*	Specifies the level of the date category *objCat*. This indicates the granularity of the rate data, such as daily or monthly. *objLevel* can be the object name or object identifier. The level is mandatory.
currencyrecordopts	Optional parameters that describe the currency in greater detail. Some options are set by default if you do not set them. For the complete list of options, see currencyrecordopts.
CurrencyRateList	Indicates the start of conversion rates.
EffectiveDate {*objCat*|0}	Specifies conversion rates for the currency. This argument is repeated for each time period that has a conversion rate. Each conversion rate specification starts with the keyword `EffectiveDate`. *objCat* indicates the date for each conversion rate; it is the date category and can be specified by object name, object identifier, or both. Zero (`0`) indicates that the country is part of the European Monetary Union (EMU).
ConversionRate *rate*	*rate* is the exchange rate for that date. It can include decimals; for example, 3.42556.

Argument	Description
RateIsBySource {**True**|**False**}	`RateIsBySource` indicates whether the conversion rate is sourced from an external data source or was manually entered. The default is `False`.
CurrencyTableType *tabletype*	`CurrencyTableType` specifies the table type that applies to this currency record for the time period. *tabletype* can be one of `BaseTable`, `EuroTable`, or `OtherTable`. The default is `BaseTable` unless `EffectiveDate` is followed by zero (`0`), in which case the default is `EuroTable`.

Example

This example adds a new currency to the currency table, and specifies conversion rates for three months.

```
CurrencyAdd "C$" Dimension "Years"
Levels "Month" CountryCode "Can"

CurrencyRateList EffectiveDate "199601"
ConversionRate 1.2

RateIsBySource False EffectiveDate "199602"
ConversionRate 1.75

RateIsBySource False EffectiveDate "199603"ConversionRate 1.55

RateIsBySource False EndList
```

CurrencyDelete

The `CurrencyDelete` verb removes a currency from the currency table.

The Windows interface equivalent is to manually remove a currency from the currency table.

The syntax is as follows:

`CurrencyDelete` *`objCurrency`*

Argument	Description
CurrencyDelete *objCurrency*	Deletes the currency *objCurrency*. *objCurrency* can be the object name, object identifier, or both.

Example

This example deletes Canadian Dollars (C$) from the currency table.

```
CurrencyDelete "C$"
```

CurrencyMake

The `CurrencyMake` verb adds a currency to the currency table or updates an existing currency.

The Windows interface equivalent is to modify the property sheet for an existing currency. For a new currency, right-click a currency in the **Currency Table** property sheet and then select **Add New Currency**.

A currency table must exist before you can create currencies. To create a currency table, see "CurrencyTableMake" (p. 133).

Currency conversions are only applied to measures if the measure definition contains the option `IsCurrency True`.

Note: Category object names differ between MDL and the Windows interface. MDL uses the category code as the object name. For more information, see "Locating Objects Uniquely".

The syntax is as follows:

```
CurrencyMake objCurrency [
Dimension objDim]
 [Drill objDrillCat]
Levels objLevel
[currencyrecordopts] CurrencyRateListEffectiveDate
{objCat|0} ConversionRate rate  [RateIsBySource {True|False}]
[CurrencyTableType tabletype]
EndList
```

Argument	Description
CurrencyMake *objCurrency*	Adds the currency *objCurrency* to the list of currencies in the currency table or modifies it if it exists. *objCurrency* can be the object name, object identifier, or both. Include the object name if the currency does not exist.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the level *objLevel*. *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Is specified, if necessary, to uniquely identify the date level *objLevel*. *objDrillCat* can be the drill category object name, object identifier, or both.
Levels *objLevel*	Specifies the level of the date category *objCat*. This indicates the granularity of the rate data, such as daily or monthly. *objLevel* can be the object name or object identifier. The level is mandatory.

Argument	Description
currencyrecordopts	Optional parameters that describe the currency in greater detail. Some options are set by default if you do not set them. For the complete list of options, see currencyrecordopts. If the currency exists, previously set options are retained unless you change them with this command.
CurrencyRateList	Indicates the start of conversion rates.
EffectiveDate {*objCat*|0}	Specifies conversion rates for the currency. This argument is repeated for each time period that has a conversion rate. Each conversion rate specification starts with the keyword `EffectiveDate`. *objCat* indicates the date for each conversion rate; it is the date category and can be specified by object name, object identifier, or both. Zero (`0`) indicates that the country is part of the European Monetary Union (EMU).
ConversionRate *rate*	*rate* is the exchange rate for that date. It can include decimals; for example, 3.42556.
RateIsBySource {**True**|**False**}	`RateIsBySource` indicates whether each rate is sourced from the external data source or was manually entered. The default is `False`.
CurrencyTableType *tabletype*	`CurrencyTableType` specifies the table type that applies to the currency record for the time period. *tabletype* can be one of `BaseTable`, `EuroTable`, or `OtherTable`. The default is `BaseTable` unless `EffectiveDate` is followed by zero (`0`), in which case the default is `EuroTable`.

Example

This example defines a Canadian currency for the currency table. It provides conversion rates for 1996 only, so the rates for 1997 are set to the default, 1.00000.

```
CurrencyMake "Canadian Dollars" Dimension "Years" Drill "By Time" Levels
"Month" CountryCode "CAN" CurrencyCountryLabel "Canada"
CurrencyFormatOverride False CurrencySymbol "$" CurrencyDecimals
2 CurrencyRateList EffectiveDate "19960101-19960131" ConversionRate 1.
36545 RateIsBySource False CurrencyTableType BaseTable EffectiveDate "19960201-
19960229" ConversionRate 1.37524 RateIsBySource False
```

```
CurrencyTableType BaseTable EffectiveDate "19960301-19960331" ConversionRate
1.36476 RateIsBySource False CurrencyTableType BaseTable EffectiveDate "19960401-
19960430" ConversionRate 1.35955 RateIsBySource False
CurrencyTableType BaseTable EffectiveDate "19960501-19960531" ConversionRate
1.36957 RateIsBySource False CurrencyTableType BaseTable EffectiveDate "19960601-
19960630" ConversionRate 1.367 RateIsBySource False
CurrencyTableType BaseTable EffectiveDate "19960701-19960731" ConversionRate
1.36826 RateIsBySource False CurrencyTableType BaseTable EffectiveDate "19960801-
19960831" ConversionRate 1.37095 RateIsBySource False
CurrencyTableType BaseTable EffectiveDate "19960901-19960930" ConversionRate
1.369 RateIsBySource False CurrencyTableType BaseTable EffectiveDate "19961001-
19961031" ConversionRate 1.34955 RateIsBySource False
CurrencyTableType BaseTable EffectiveDate "19961101-19961130" ConversionRate
1.3375 RateIsBySource False CurrencyTableType BaseTable EffectiveDate "19961201-
19961231" ConversionRate 1.3635 RateIsBySource False CurrencyTableType BaseTable
```

CurrencyTableAdd

The `CurrencyTableAdd` verb creates a currency table in your Transformer model.

The Windows interface equivalent is to select **Currency Table** from the **File** menu.

If you specify the option `CurrencyTableType`, the object name of the currency table is optional.

For more information about creating Currency Tables, see "CurrencyTableMake" (p. 133).

The syntax is as follows:

CurrencyTableAdd [*objCurrencyTable*]
[*currencytableopts*] **Associations** *assocopts*

Argument	Description
CurrencyTableAdd *objCurrencyTable*	Creates the currency table *objCurrencyTable*. *objCurrencyTable* can be the object name, object identifier, or both. Include the name for clarity in your model.
currencytableopts	Optional parameters that specify columns in an external data source. For the complete list of options, see currencytableopts. If the currency exists, previously set options are retained unless you change them with this command.
Associations *assocopts*	Required parameters that specify associations for the currency table. For the complete list of options, see assocopts.

Example

This example creates the base currency table.

```
CurrencyTableAdd 869 "CurrencyBaseTable" CurrencyTableType BaseTable
Associations 1315 "Country Code" AssociationType Type_Query AssociationRole
Role_CountryCode
```

```
AssociationReferenced "Country Code"
Associations 1317 "Date" AssociationType Type_Query AssociationRole Role_
Date AssociationReferenced "Date"
Associations 1319 "Currency" AssociationType Type_Query AssociationRole
Role_Label
AssociationReferenced "Currency"
Associations 1321 "Conversion Rate" AssociationType Type_Query
AssociationRole Role_Rate
AssociationReferenced "Conversion Rate"
```

CurrencyTableDelete

The `CurrencyTableDelete` verb removes an existing currency table from the model.

The Windows interface equivalent is the **Remove Currency Table** option on the **File** menu.

The syntax is as follows:

`CurrencyTableDelete` *`objCurrencyTable`*

Argument	Description
CurrencyTableDelete *objCurrencyTable*	Deletes the currency table *objCurrencyTable*. *objCurrencyTable* can be the object name, object identifier, or both.

Example

This example deletes the base currency table from the Transformer model.

```
CurrencyTableDelete "CurrencyBaseTable"
```

CurrencyTableMake

The `CurrencyTableMake` verb creates a currency table or updates an existing currency table.

The Windows interface equivalent is to modify the properties on the **Currency Table** property sheet.

On the Transformer version 8.x interface, the currency table has a single property sheet, even when Euro triangulation is in effect. In MDL, the Base Currency and Euro Currency tables appear as two separate objects. Each has its own `CurrencyTableMake` statement.

The default object names for the two types of table are `CurrencyBaseTable` and `CurrencyEuroTable`. You can specify a different object name, but on the Windows interface, the default object always appears.

If you specify the option `CurrencyTableType`, the object name of the currency table is optional.

The syntax is as follows:

`CurrencyTableMake` [*`objCurrencyTable`*]
[*`currencytableopts`*] **`Associations`** *`assocopts`*

Argument	Description
CurrencyTableMake *objCurrencyTable*	Creates the currency table *objCurrencyTable* or modifies it if it exists. *objCurrencyTable* can be the object name, object identifier, or both. Include the object name if the currency table does not exist.
currencytableopts	Optional parameters that specify columns in an external data source. For the complete list of options, see currencytableopts. If the currency exists, previously set options are retained unless you change them with this command.
Associations *assocopts*	Optional parameters that specify associations for the currency table. For the complete list of options, see assocopts.

Example

This example defines a currency table.

```
CurrencyTableMake 869 "CurrencyBaseTable" CurrencyTableType BaseTable
Associations 1315 "Country Code" AssociationType Type_Query AssociationRole
Role_CountryCode
AssociationReferenced "Country Code"
Associations 1317 "Date" AssociationType Type_Query AssociationRole Role_
Date AssociationReferenced "Date"
Associations 1319 "Currency" AssociationType Type_Query AssociationRole
Role_Label
AssociationReferenced "Currency"
Associations 1321 "Conversion Rate" AssociationType Type_Query
AssociationRole Role_Rate
AssociationReferenced "Conversion Rate"
```

CurrencyTableUpdate

The `CurrencyTableUpdate` verb updates an existing currency table in the model.

The Windows interface equivalent is to modify the properties on the **Currency Table** property sheet.

If you specify the option `CurrencyTableType`, the object name of the currency table is optional.

For more information about updating Currency Tables, see "CurrencyTableMake" (p. 133).

The syntax is as follows:

`CurrencyTableUpdate` [*objCurrencyTable*]
[*currencyopts*] **`Associations`** *assocopts*

Argument	Description
CurrencyTableUpdate *objCurrencyTable*	Updates the currency table *objCurrencyTable*. *objCurrencyTable* can be the object name, object identifier, or both.
currencytableopts	Optional parameters that specify columns in an external data source. For the complete list of options, see currencytableopts. If the currency exists, previously set options are retained unless you change them with this command.
Associations *assocopts*	Optional parameters that specify associations for the currency table. For the complete list of options, see assocopts.

Example

This example modifies the currency table by changing its data source, which requires specifying different columns.

```
CurrencyTableMake 869 "Currency Base Table" CurrencyTableType BaseTable
Associations 1315 "Country Code" AssociationType Type_Query AssociationRole
Role_CountryCode
AssociationReferenced "Country Code"
Associations 1317 "Date" AssociationType Type_Query AssociationRole Role_Date
AssociationReferenced "Date"
Associations 1319 "Currency" AssociationType Type_Query AssociationRole
Role_Label
AssociationReferenced "Currency"
Associations 1321 "Conversion Rate" AssociationType Type_
Query AssociationRole Role_Rate
AssociationReferenced "Conversion Rate"
```

CurrencyUpdate

The `CurrencyUpdate` verb updates an existing currency in a currency table.

The Windows interface equivalent is to modify the properties on the **Currency Table** property sheet.

For more information about updating currencies, see "CurrencyMake" (p. 130).

The syntax is as follows:

```
CurrencyUpdate objCurrency [
Dimension objDim]
[Drill objDrillCat]
Levels objLevel [
currencyrecordopts] [CurrencyRateListEffectiveDate
{objCat|0} ConversionRate rate [RateIsBySource {True|False}]
[CurrencyTableType tabletype]
EndList
```

Argument	Description
CurrencyUpdate *objCurrency*	Modifies the currency *objCurrency*. *objCurrency* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the level *objLevel*. *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Is specified, if necessary, to uniquely identify the date level *objLevel*. *objDrillCat* can be the drill category object name, object identifier, or both.
Levels *objLevel*	Specifies the level of the date category *objCat*. This indicates the granularity of the rate data, such as daily or monthly. *objLevel* can be the object name or object identifier. The level is mandatory.
currencyrecordopts	Optional parameters that describe the currency in greater detail. For the complete list of options, see currencyrecordopts. Previously set options are retained unless you change them with this command.
CurrencyRateList	Indicates the start of conversion rates.
EffectiveDate {*objCat*|0}	Specifies conversion rates for the currency. This argument is repeated for each time period that has a conversion rate. Each conversion rate specification starts with the keyword `EffectiveDate`. *objCat* indicates the date for each conversion rate; it is the date category and can be specified by object name, object identifier, or both. Zero (`0`) indicates that the country is part of the European Monetary Union (EMU).
ConversionRate *rate*	*rate* is the exchange rate for that date. It can include decimals; for example, 3.42556.

Argument	Description
RateIsBySource {**True**|**False**}	`RateIsBySource` indicates whether each rate is sourced from the external data source or was manually entered. The default is `False`.
CurrencyTableType *tabletype*	`CurrencyTableType` specifies the table type that applies to the currency record for the time period. *tabletype* can be one of `BaseTable`, `EuroTable`, or `OtherTable`. The default is `BaseTable` unless `EffectiveDate` is followed by zero (`0`), in which case the default is `EuroTable`.

Example

This example modifies the conversion rate for Canadian Dollars for December 1997.

```
CurrencyUpdate "Canadian Dollars" CurrencyRateList EffectiveDate "19971201-
19971231" ConversionRate 1.55 EndList
```

CustomViewAdd

The `CustomViewAdd` verb creates a custom view in the model.

The Windows interface equivalent for creating a custom view is to select the **Create Custom View** command in the diagram.

The syntax is as follows:

CustomViewAdd"*customview*"**DimensionView**"*All Categories*" **DimensionView** *objView* **MeasureInclude** *objMeasure* {Yes|No}

Argument	Description
CustomViewAdd "*custom view*"	Adds the custom view "*custom view*". "*custom view*" must be the object name and can include the object identifier.
DimensionView "*All Categories*"	Specifies that all categories are initially included in the custom view.
DimensionView *objView*	Specifies the dimension and dimension view. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views.

Argument	Description
MeasureInclude *objMeasure*	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. Repeat this argument to specify multiple measures, as in `MeasureInclude 159 Yes MeasureInclude 175 No`

Example

```
CustomViewAdd "Authors" DimensionView 149 "All Categories"
DimensionView
195 "Authors~User View" MeasureInclude 225 Yes
```

CustomViewChildListUpdate

The `CustomViewChildListUpdate` verb defines the list of descendant custom views. You can only update the list of descentant custom views in MDL.

The syntax is as follows:

`CustomViewChildListUpdate` *custom view ID* **`StartList`** *custom view ID* **`Endlist`**

Examples

```
CustomViewChildListUpdate 11529 StartList 11537 Endlist
CustomViewChildListUpdate 11537 StartList EndList
```

CustomViewDelete

The `CustomViewDelete` verb deletes a custom view from the model.

The Windows interface equivalent for deleting a custom view is to right-click the custom view, and then click **Delete**.

The syntax is as follows:

`CustomViewDelete` *"customview"*

Argument	Description
CustomViewDelete "*custom view*"	Deletes the custom view "*custom view*". "*custom view*" must be the object name and can include the object identifier.

Example

```
CustomViewDelete "Authors"
```

CustomViewMake

The `CustomViewMake` verb creates a custom view in the model.

The Windows interface equivalent for creating a custom view is to select the **Create Custom View** command in the diagram.

The syntax is as follows:

CustomViewMake *"customview"* **DimensionView** *objView* **MeasureInclude** *objMeasure* {Yes|No}

Argument	Description
CustomViewMake "*custom view*"	Adds the custom view "*custom view*". "*custom view*" must be the object name and can include the object identifier.
DimensionView *objView*	Specifies the dimension and dimension view. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views.
MeasureInclude *objMeasure* {Yes|No}	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. Use Yes to include the measure or No to exclude it. Repeat this argument to specify multiple measures, as in `MeasureInclude 159 Yes MeasureInclude 175 No`

Example

```
CustomViewMake "Authors" DimensionView 149 "All Categories"
DimensionView
195 "Authors~User View" MeasureInclude 225 Yes
```

CustomViewUpdate

The `CustomViewUpdate` verb modifies a custom view in the model.

The Windows interface equivalent for creating a custom view is to right-click the custom view, and then modify the custom view properties.

The syntax is as follows:

CustomViewUpdate *"customview"* **DimensionView** *objView* **MeasureInclude** *objMeasure* {Yes|No}

Argument	Description
CustomViewUpdate "*custom view*"	Adds the data source "*custom view*". "*custom view*" must be the object name and can include the object identifier.
DimensionView *objView*	Specifies the dimension and dimension view. *objView* can be the object name, object identifier, or both. Repeat this argument to specify multiple views.
MeasureInclude *objMeasure* {Yes|No}	Specifies the measure. *objMeasure* can be the object name, object identifier, or both. Use Yes to include the measure or No to exclude it. Repeat this argument to specify multiple measures, as in `MeasureInclude 159 Yes MeasureInclude 175 No`

Example

```
CustomViewUpdate "Authors" DimensionView 149 "All Categories"
DimensionView
195 "Authors~User View" MeasureInclude 225 No
```

DataSourceAdd

The `DataSourceAdd` verb creates a data source other than a package or report in the model. This verb is not used to create a package or report data source. For information about creating a package or report data source, see "CognosPackageAdd" (p. 101).

The Windows interface equivalent for creating a data source is to select the **Insert** command on the **Edit** menu when a data source is selected.

The Windows interface equivalent for creating a query based on an IBM Cognos 8 package or report data source is to select the **Add Query From Package or Report** command on the **Edit** menu when a data source is selected.

Notes:

- If you include a data source ID for the new data source in the syntax, and the object already exists, you will receive an error message.
- To insert a dimension from a package (the **Insert Dimension From Package** command on the **Edit** menu, you must first add the data source and then add a dimension from the data source. For more information about adding a dimension from a data source, see "DimAdd" (p. 145).

For more information about creating data sources, see "DataSourceMake" (p. 142).

The syntax for a query based on an IBM Cognos 8 package or report is as follows:

```
DataSourceAdd objDataSource Sourcetype CognosSourceQuery PackageReportSource
sourceID
"source name" [appqueryopts] [OrgName objCol Origin {Source|Generated|Calculated|
Manual}Offset offset][
colopts]
```

The syntax for all other data source types is as follows:

```
DataSourceAdd objDataSource [appqueryopts]
[OrgName objCol Origin
{Source|Generated|Calculated|Manual}Offset offset]
[colopts]
```

Argument	Description
DataSourceAdd *objDataSource*	Creates the data source *objDataSource*. *objDataSource* must be the object name and can include the object identifier.
appqueryopts	Optional parameters that describe the data source in greater detail. Some options are set by default if you do not set them. For the complete list of options, see appqueryopts.
OrgName *objCol* **Origin {Source|Generated|Calculated|Manual} Offset** *offset*	Specifies the original name of the column, the origin and the offset. Origin must be one of `Source`, `Generated`, `Calculated`, or `Manual`. *offset* can be `0` or greater. This allows you to take the structured definition of the column and append it to the data source definition. For more information about structured format, see "Structured MDL".
colopts	Optional parameters that describe the columns in greater detail. Some options are set by default if you do not set them. For the complete list of options, see colopts.

Examples

This example adds a query based on the IBM Cognos 8 package data source Go Sales and Retailers to the model.

```
DataSourceAdd "GOSRQuery" SourceType CognosSourceQuery
PackageReportSource
103 "GO Sales and Retailers" Separator "," CharacterSet Default
DecimalSep "." Thousandsep "," Columns True Timing PopYesCreateDefault
AutoSummary
True SetCurrent True ServerSource False Speed False Presummarized
False
```

This example adds the data source Products (CSV) to the model.

```
DataSourceAdd "Products (CSV)" Source "c:\prodinfo.csv" SourceType
FlatFileColNames
```

DataSourceDelete

The `DataSourceDelete` verb removes a data source other than a package or report from the model. This verb does not delete package or report data sources. For information about deleting package and report data sources, see "CognosPackageDelete" (p. 102).

The Windows interface equivalent is to click **Delete** on the **Edit** menu when a data source is selected.

For more information about IBM Cognos 8 data sources, see the Transformer *User Guide* and the Framework Manager *User Guide*.

The syntax is as follows:

DataSourceDelete *objDataSource*

Argument	Description
DataSourceDelete *objDataSource*	Deletes the data source *objDataSource*. *objDataSource* can be the object name, object identifier, or both.

Example

This example deletes the All Staff Count (Excel) data source.

```
DataSourceDelete "All Staff Count (Excel)"
```

DataSourceMake

The `DataSourceMake` verb creates a data source other than a package or report or updates an existing one. This verb does not create package or report data sources. For information about creating package and report data sources, see "CognosPackageMake" (p. 102).

The Windows interface equivalent, if the data source exists, is to modify the **Data Source** property sheet. For a new data source, when the data source is selected, the equivalent is to select the **Insert** command on the **Edit** menu.

The syntax for a query based on an IBM Cognos 8 package or report is:

```
DataSourceMake data
source ID objDataSource SourceType CognosSourceQuery PackageReportSource source
 ID "source name"  [
appqueryopts] [OrgName objCol Origin {Source|Generated|Calculated|Manual}
Offset offset] [
colopts]
```

The syntax for all other data source types is as follows:

```
DataSourceMake datasource
ID objDataSource [appqueryopts]
[OrgName objCol Origin
{Source|Generated|Calculated|Manual}Offset offset]
[colopts]
```

Argument	Description
DataSourceMake *objDataSource*	Creates the data source *objDataSource* or modifies it if it exists. *objDataSource* can be the object name, object identifier, or both. Include the object name if the category does not exist.
appqueryopts	Optional parameters that describe the data source in greater detail. Some options are set by default if you do not set them. For the complete list of options, see appqueryopts. If the data source exists, previously set options are retained unless you change them with this command.
OrgName *objCol* **Origin** {**Source**|**Generated**|**Calculated**|**Manual**} **Offset** *offset*	Specifies the original name of the column, the origin, and the offset. Origin must be one of `Source`, `Generated`, `Calculated`, or `Manual`. *offset* can be `0` or greater. This allows you to take the structured definition of the column and append it to the data source definition. For more information on structured format, see "Structured MDL".
colopts	Optional parameters that can describe the columns in greater detail. For the complete list of options, see colopts.

Examples

This example adds the data source Go Sales and Retailers to the model.

```
DataSourceMake 103 "Go Sales and Retailers" SourceType
Package SourcePath "/content/package[@name='GO Sales and Retailers']"
PackageTimeStamp "/content/package[@name='GO Sales and Retailers200701011200']/
model[@name='model']"
```

This example adds a query based on the data source Go Sales and Retailers to the model.

```
DataSourceMake 105 "Go Sales and Retailers~1" SourceType
CognosSourceQuery
PackageReportSource 103 "GO Sales and Retailers" Separator "," CharacterSet
Default DecimalSep "." Thousandsep "," Columns True Timing PopYesCreateDefault
AutoSummary True SetCurrent True ServerSource False Speed False
Presummarized False
```

This example defines the data source All Staff Count.

```
DataSourceMake 106 "Products (CSV)"  Separator "," SourceType FlatFile_
ColNames CharacterSet Multibyte DecimalSep "." Thousandsep "," Columns False
 Timing PopYesCreateDefault Source "prodinfo.csv"
```

DataSourceUpdate

The `DataSourceUpdate` verb updates an existing data source other than a package or report in the model. This verb does not update package or report data sources. For information about creating package and report data sources, see "CognosPackageUpdate" (p. 103)

The Windows interface equivalent is to modify the **Data Source** property sheet.

Note: If you include a data source ID for the updated data source in the syntax, and the object does not already exist, you will receive an error message.

The syntax is as follows:

```
DataSourceUpdate objDataSource [
appqueryopts]
```

Argument	Description
DataSourceUpdate *objDataSource*	Specifies the data source to update. *objDataSource* can be the object name, object identifier, or both.
appqueryopts	Optional parameters that describe the data source in greater detail. For the complete list of options, see appqueryopts. Previously set options are retained unless you change them with this command.

Example

This example disables AutoSummary for the data source Go Sales and Retailers~1.

```
DataSourceUpdate 105 "Go Sales and Retailers~1" Separator
"," CharacterSet Default DecimalSep "." Thousandsep "," Columns
True Timing PopYesCreateDefault AutoSummary True SetCurrent True
ServerSource False Speed False Presummarized False
```

This example changes the source file for the data source All Staff Count (Excel) to bigstaff.asc, and changes the file format to ASCII.

```
DataSourceUpdate "All Staff Count (Excel)" Source "c:installation_directory\
bigstaff.asc" SourceType FlatFile_ColNames
```

DeletionListUpdate

The `DeletionListUpdate` verb updates the list of objects to be removed from the model.

The Windows interface equivalent in Transformer version 8.x is to delete objects from the model by means of an incremental update.

We strongly recommend that you do not change this syntax in generated MDL, because this action could cause unexpected results in your model.

The syntax is as follows:

```
DeletionListUpdate deletionsoptsEndlist
```

Argument	Description
DeletionListUpdate *deletionsopts*	Deletes an object from the model by referring to its type and object identifier. *deletionsopts* are required parameters that describe the object to be deleted. For the complete list of options, see deletion-sopts.

Example

This example deletes the category with object identifier 1001.

```
DeletionListUpdate Category 1001 Endlist
```

DimAdd

The `DimAdd` verb creates a dimension in the model.

The Windows equivalent is to select the **Insert** command on the **Edit** menu when a dimension is selected.

For more information about creating dimensions, see "DimMake" (p. 151).

The syntax is as follows:

```
DimAdd objDim [
dimopts] Associations assocopts
```

Argument	Description
DimAdd *objDim*	Creates the dimension *objDim*. *objDim* must be the object name and can include the object identifier.
dimopts	Optional parameters that describe the dimension in greater detail. Some options are set by default if you do not set them. For the complete list of options, see dimopts.
Associations *assocopts*	Parameters that specify associations for the dimension. For the complete list of options, see assocopts.

Example

This example creates the Years dimension.

```
DimAdd 231 "Years" DimType Date EarliestDate 19000101 LatestDate 99991231
ManualPeriods False DaysInWeek 127 NewCatsLock
False ExcludeAutoPartitioning False
```

```
Associations 1249 "Time Category Code" AssociationType Type_Query
AssociationRole Role_Source AssociationReferenced "Time Category Code
```

DimCalcDefAdd

The `DimCalcDefAdd` verb creates a dimension calculation definition in the model.

The Windows interface equivalent is the **Add** command on the **Calculation** tab of the **Dimension** property sheet.

For more information about creating dimension calculation definitions, see "DimCalcDef-Make" (p. 148).

The syntax is as follows:

```
DimCalcDefAdd objDimCalcDefDimension objDim [
Calc expropts] [
GroupCalculateCategory {True|False}]
[Set stringStartList objCatsEndList]
```

Argument	Description
DimCalcDefAdd *objDimCalcDef*	Creates the dimension calculation definition *objDimCalcDef*. *objDimCalcDef* must be the object name and can also be the object identifier.
Dimension *objDim*	Specifies the dimension where the calculated category is to be located. *objDim* can the object name, object identifier, or both.
Calc *expropts*	Specifies the calculation. *expropts* are a mixture of calculation keywords and objects. The calculation keywords that can be used are `Average`, `Max`, and `Min`, as well as the Transformer-specific functions `Change`, `Percent-Growth` and `Share`. The objects are specified as object name followed by an at sign (@)), the object type, and, optionally, an at sign (@)) and the object identifier. The object type can be `Category`, `Level`, or `Drill`. For example, `"Go Water Bottle@Category@4805"`.

Argument	Description
GroupCalculateCategory {**True**|**False**}	Is an option specifying that calculated categories be grouped together in the dimension viewer. The equivalent in the Windows interface is the **Group Calculated Categories Together** check box, which appears in the **Category Calculation Definition** dialog box when you create a category definition. The default is `False`.

Set *string* **StartList** *objCats*	Creates one or more sets and specifies the categories in each set. *string* is the set name, such as `Set 1`, `Set 2`, and so on. *objCats* can be the object name or the object identifier. This argument is repeated for each set; for example
	`Set "Set 1" StartList 4795 4797 EndList`
	`Set "Set 2" StartList 4799 EndList`

Example

This example adds a share calculation in the Productions dimension. The statement creates two objects: the dimension calculation definition and an associated category.

```
DimCalcDefAdd 'share ("Set 1", "GO Sport Line")' Dimension 2947 Calc share
("Set 1@Set", "GO Sport Line@Category@4789") Set "Set 1" StartList 4793 4797
4801 4803 EndList
```

DimCalcDefDelete

The `DimCalcDefDelete` verb removes a dimension calculation definition from the model.

The Windows interface equivalent is the **Remove** command on the **Calculation** tab of the **Dimension** property sheet.

The syntax is as follows:

DimCalcDefDelete *objDimCalcDef* [
Dimension *objDim*]

Argument	Description
DimCalcDefDelete *objDimCalcDef*	Deletes the dimension calculation definition *objDimCalcDef*. *objDimCalcDef* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify *objDimCalcDef*. *objDim* can be the object name, object identifier, or both.

Example

This example deletes the dimension calculation definition that uses object identifier 27195.

```
DimCalcDefDelete 27195
```

DimCalcDefMake

The `DimCalcDefMake` verb creates or updates a dimension calculation definition in the model.

The Windows interface equivalents are the **Add** and **Modify** commands on the **Calculation** tab of the **Dimension** property sheet.

There are two parts to a dimension calculation definition in MDL: a `DimCalcDefMake` or `DimCalcDefAdd` statement defining the calculation, and a `CatMake` or `CatAdd` statement defining the category.

The link between the two is an option in the `CatMake` or `CatAdd` statement that references the `DimCalcDefMake` or `DimCalcDefAdd` statement. For more information, see "DimCalcDefAdd" (p. 146).

Calculated categories are defined as regular categories, not special categories.

The syntax is as follows:

```
DimCalcDefMake objDimCalcDef [
Dimension objDim]
GroupCalculateCategory {True|False}]
[Calc expropts]
[Set stringStartList objCatsEndList]
```

Argument	Description
DimCalcDefMake *objDimCalcDef*	Creates the dimension calculation definition *objDimCalcDef* or modifies it if it exists. Include the object name if the definition does not exist.
Dimension *objDim*	Is specified, if necessary, to uniquely identify *objDimCalcDef*. *objDim* can be the object name, object identifier, or both.
GroupCalculateCategory {**True**|**False**}	Specifies that calculated categories be grouped together in the dimension viewer. The equivalent in the Windows interface is the **Group Calculated Categories Together** check box, which appears in the **Category Calculation Definition** dialog box when you create a category definition. The default is `False`.

Argument	Description
Calc *expropts*	Specifies the calculation. *expropts* are a mixture of calculation keywords and objects. The calculation keywords that can be used are `Average`, `Max`, and `Min`, as well as the Transformer-specific functions `Change`, `Percent-Growth` and `Share`. The objects are specified as object name followed by an at sign (`@`), the object type, and, optionally, an at sign (`@`) and the object identifier. The object type can be `Category`, `Level`, or `Drill`. For example, `"Go Water Bottle@Category@4805"`.

Sets *tring* **StartList** *objCats* **Endlist**	Creates one or more sets and specifies the categories in each set. *string* is the set name, such as `Set 1`, `Set 2`, and so on. *objCats* can be the object name or the object identifier. This argument is repeated for each set; for example
	`Set "Set 1" StartList 4795 4797 EndList`
	`Set "Set 2" StartList 4799 EndList`

Example

This example defines the dimension calculation Share ("Set 1", "GO Sport Line").

```
DimCalcDefMake 27195 'share("Set 1", "GO Sport Line")' Dimension 2947
GroupCalculateCategory True Calc share ("Set 1@Set", "GO Sport
Line@Category@4789") Set "Set 1" StartList 4793 4797 4801 4803 EndList
```

DimCalcDefUpdate

The `DimCalcDefUpdate` verb modifies a dimension calculation definition in the model.

The Windows interface equivalent is the **Modify** command on the **Calculation** tab of the **Dimension** property sheet.

For more information about updating dimension calculation definitions, see "DimCalcDefMake" (p. 148).

The syntax is as follows:

```
DimCalcDefUpdate objDimCalcDef [
Dimension objDim] [
GroupCalculateCategory {True|False}]
[Calc expropts]
[Set string StartList objCatsEndList]
```

Argument	Description
DimCalcDefUpdate *objDimCalcDef*	Modifies the dimension calculation definition *objDimCalcDef*. *objDimCalcDef* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify *objDimCalcDef*. *objDim* can be the object name, object identifier, or both.
GroupCalculateCategory {**True**|**False**}	Is an option specifying that calculated categories be grouped together in the dimension viewer. The equivalent in the Windows interface is the **Group Calculated Categories Together** check box, which appears in the **Category Calculation Definition** dialog box when you create a category definition. The default is `False`.
Calc *expropts*	Specifies the calculation. *expropts* are mixture of calculation keywords and objects. The calculation keywords that can be used are `Average`, `Max`, and `Min`, as well as the Transformer-specific functions `Change`, `Percent-Growth` and `Share`. The objects are specified as object name followed by an at sign `(@)`, the object type, and, optionally, an at sign `(@)` and the object identifier. The object type can be `Category`, `Level`, or `Drill`. For example, `"Go Water Bottle@Category@4805"`.

Set *string* **StartList** *objCats* **EndList**	Creates one or more sets and specifies the categories in each set. *string* is the set name, such as `Set 1`, `Set 2`, and so on. *objCats* can be the object name or the object identifier. This argument is repeated for each set; for example
	`Set "Set 1" StartList 4795 4797 EndList`
	`Set "Set 2" StartList 4799 EndList`

Example

This example modifies dimension calculation Share ("Set 1", "GO Sport Line") by adding the category with object identifier 4789 into Set 1.

```
DimCalcDefUpdate 27195 'share("Set 1", "GO Sport Line")' Dimension 2947 Calc
share("Set 1@Set", "GO Sport Line@Category@4789") Set "Set 1" StartList 4793
4795 4797 4801 4803 EndList
```

DimDelete

The `DimDelete` verb removes a dimension from the model.

The Windows interface equivalent is to click **Delete** on the **Edit** menu when a dimension is selected.

The syntax is as follows:

`DimDelete` *`objDim`*

Argument	Description
DimDelete *objDim*	Deletes the dimension *objDim*. *objDim* can be the object name, object identifier, or both.

Example

This example deletes the dimension Products.

```
DimDelete "Products"
```

DimensionListUpdate

The `DimensionListUpdate` verb reorders the list of dimensions in the model.

The Windows interface equivalent is to drag and drop dimensions into the required positions.

The syntax is as follows:

`DimensionListUpdate` *`objDims`* **`EndList`**

Argument	Description
DimensionListUpdate *objDims*	Specifies the dimensions in the desired order. If not all dimensions are specified, the specified ones move to the top of the list. The list of dimensions can be object names or object identifiers.

Example

This example reorders the list so that Products and Years appear as the first two dimensions.

```
DimensionListUpdate "Products" "Years" EndList
```

DimMake

The `DimMake` verb creates a dimension or updates an existing one.

The Windows interface equivalent, if the dimension exists, is to modify the **Dimension** property sheet. For a new dimension, select the **Insert** option on the **Edit** menu when a dimension is selected.

Dimensions require a root category, a drill category and two default dimension views. These are created automatically when a dimension is added on the Windows interface, but they are only created in MDL if you use the verb `ModelEnsureCompleteness`.

The syntax is as follows:

```
DimMake objDim [
dimopts] Associations assocopts
```

Argument	Description
DimMake *objDim*	Creates the dimension *objDim* or modifies it if it exists. *objDim* can be the object name, object identifier, or both. Include the object name if the category does not exist.
dimopts	Optional parameters that describe the dimension in greater detail. Some options are set by default if you do not set them. For the complete list of options, see dimopts. If the dimension exists, previously set options are retained unless you change them with this command.
Associations *assocopts*	Parameters that specify associations for the dimension. For the complete list of options, see assocopts.

Example

This example defines the dimension Products.

```
DimMake "Products" DimType Regular NewCatsLock False DimInfo "Products
carried by Great Outdoors Company"
```

DimUpdate

The `DimUpdate` verb updates an existing dimension.

The Windows interface equivalent is to modify the **Dimension** property sheet.

For more information about updating dimensions, see "DimMake" (p. 151).

The syntax is as follows:

```
DimUpdate objDim [
dimopts] [Associationsassocopts]
```

Argument	Description
DimUpdate *objDim*	Specifies the dimension to update. *objDim* can be the object name, object identifier, or both.

Argument	Description
dimopts	Optional parameters that describe the dimension in greater detail. For the complete list of options, see dimopts. Previously set options are retained unless you change them with this command.
Associations *assocopts*	Optional parameters that specify associations for the dimension. For the complete list of options, see assocopts.

Example

This example changes the earliest and latest dates allowed in the Years dimension.

```
DimUpdate "Years" EarliestDate 19900101 LatestDate 20100101
```

DrillCatMake

The `DrillCatMake` verb creates or updates a drill category.

The Windows interface equivalent is to modify the **Drill Category** property sheet. Drill categories are created automatically when you build models on the Windows interface.

The syntax is as follows:

```
DrillCatMake objDrillCat [
Dimension objDim]
[Root objRootCat][
JoiningLevel objLevel]
[catopts]
```

Argument	Description
DrillCatMake *objDrillCat*	Creates the drill category *objDrillCat* or changes it if it exists. *objDrillCat* can be the object name, object identifier, or both. Include the object name if the category does not exist.
Dimension *objDim*	When creating a drill category, this specifies its dimension. When updating a drill category this is used, if necessary, to uniquely identify it. *objDim* can be the object name, object identifier, or both.
Root *objRootCat*	When creating a drill category, this specifies its root category. *objRootCat* can be the object name, object identifier, or both.

Argument	Description
JoiningLevel *objLevel*	When creating a drill category, this specifies the convergence level. *objLevel* can be the object name or object identifier.
catopts	Optional parameters that describe the category in greater detail. Some options are set by default if you do not set them. For the complete list of options, see catopts. If the category exists, previously set options are retained unless you change them with this command.

Example

This example defines the drill category By Product Line.

```
DrillCatMake "By Product Line" Dimension "Products" Root "Product Line"
Inclusion Suppress Filtered False Suppressed True PrimaryDrill True YearBegins
 19960101 PartialWeek Split ExtraWeek None WeekBegins Sunday
```

EventEnd

The `EventEnd` verb signals the end of an update event that was started by the `EventStart` verb. The status of the PowerCubes involved in the event reverts to `Free`. There is no Windows interface equivalent for this verb.

In IBM Cognos 8, you do not specify a client/server platform type.

The syntax is as follows:

`EventEnd`

Example

This example ends the current update event.

```
EventEnd
```

EventStart

The `EventStart` verb signals the start of an update event. The status of the PowerCubes involved in the event is changed from `Free` to `UpdateInProgress`, and the event name is written to a log file. There is no Windows interface equivalent for this verb.

The log file is a file that is created whenever you run a Transformer model. It has the same root name as the Transformer model but with a .log extension.

In IBM Cognos 8, you do not specify a client/server platform type.

The syntax is as follows:

`EventStart` *`[EventName]`*

Argument	Description
EventName	The event name can be up to 256 characters in length. It cannot contain a carriage return.

Example

This example starts the Batch Update event.

```
EventStart "Batch Update"
```

FilterCat

The `FilterCat` verb excludes (or filters out) a category from the specified dimension view. The Windows interface equivalent is the **Exclude** option on the **Diagram** menu.

Using `FilterCat` on an already-excluded category removes the filter.

Note: Category object names differ between MDL and the Windows interface. MDL uses the category code as the object name. For more information, see "Locating Objects Uniquely".

The syntax is as follows:

```
FilterCat objView [
Dimension objDim]
Category objCat
```

Argument	Description
FilterCat *objView*	Specifies the dimension view containing the category to be excluded or filtered out. *objView* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the view. *objDim* can be the object name, object identifier, or both.
Category *objCat*	Specifies the category to be excluded or filtered out. *objCat* can be the object name, object identifier, or both.

Example

This example excludes the Outdoor Products category for the dimension view View1.

```
FilterCat "View1" Dimension "Products" Category "Outdoor Products"
```

LevelAdd

The `LevelAdd` verb creates a new level in the specified model dimension.

The Windows interface equivalent is to select the **Insert** option on the **Edit** menu when a level is selected.

For more information about creating levels, see "LevelMake" (p. 157).

The syntax is as follows:

```
LevelAdd objLevel [
Dimension objDim]
[Drill objDrillCat]
[Parent objLevel]
[levelopts] [DrillList objDrillCats EndList]
Associations assocopts
```

Argument	Description
LevelAdd *objLevel*	Creates the level *objLevel*. *objLevel* must be the object name and can include the object identifier.
Dimension *objDim*	Specifies a dimension for the level. *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Specifies a drill category for the level. *objDrillCat* can be the object name, object identifier, or both.
Parent *objLevel*	Specifies a parent for the level. *objLevel* can be the object name, or object identifier.
levelopts	Optional parameters that describe the level in greater detail. Some options are set by default if you do not set them. For the complete list of options, see levelopts.
DrillList *objDrillCats*	Specifies the multiple drill categories *objDrillCats*. *objDrillCats* is one or more drill categories, each identified by object name or object identifier.
Associations *assocopts*	Parameters that specify associations for the level. For the complete list of options, see assocopts.

Example

This example creates the level Year.

```
LevelAdd 237 "Year" Blanks "(blank)" Inclusion Generate DateFunction
Year Generate Need RefreshLabel False  RefreshDescription
False RefreshShortName False NewCatsLock False CatLabFormat "YYYY" Timerank
10 UniqueCategories True UniqueMove False Associations 1251
"Time" AssociationType Type_Query AssociationRole Role_
```

```
Source AssociationReferenced "Time" Associations 1253 "Time"
AssociationContext 235 AssociationType Type_Query AssociationRole Role_
OrderBy AssociationReferenced "Time" SortOrder Default SortAs Ascending
```

LevelDelete

The `LevelDelete` verb removes a level from the specified model dimension.

The Windows interface equivalent is to click **Delete** on the **Edit** menu when a level is selected.

The syntax is as follows:

```
LevelDelete objLevel [
Dimension objDim]
[Drill objDrill]
```

Argument	Description
LevelDelete *objLevel*	Deletes the level *objLevel*. *objLevel* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the level. *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Is specified, if necessary, to uniquely identify the level. *objDrillCat* can be the object name or object identifier.

Example

This example deletes the level Product Line.

```
LevelDelete "Product Line" Dimension "Products" Drill "By Product Line"
```

LevelMake

The `LevelMake` verb creates a level or updates an existing one.

The Windows interface equivalent, if the level exists, is to modify the **Level** property sheet. For a new level, select the **Insert** option on the **Edit** menu when a level is selected.

Level names cannot contain an at sign (`@`).

The syntax is as follows:

```
LevelMake objLevel [
Dimension objDim]
[Drill objDrillCat]
[Parent objLevel]
[levelopts] [DrillList objDrillCats EndList]
Associations assocopts
```

Argument	Description
LevelMake *objLevel*	Creates the level *objLevel* or modifies it if it exists. *objLevel* can be the object name, object identifier, or both. Include the object name if the category does not exist.
Dimension *objDim*	When creating a level, this specifies its dimension. When updating a level this is used, if necessary, to uniquely identify it. *objDim* can be the object name or object identifier.
Drill *objDrillCat*	When creating a level, this specifies its drill category. When updating a level this is used, if necessary, to uniquely identify it. *objDrillCat* can be the object name, object identifier, or both.
Parent *objLevel*	When creating a level, this specifies its parent. When updating a level this is used, if necessary, to uniquely identify it. *objLevel* can be the object name or object identifier.
levelopts	Optional parameters that describe the level in greater detail. Some options are set by default if you do not set them. For the complete list of options, see levelopts. If the category exists, previously set options are retained unless you change them with this command.
DrillList *objDrillCats*	Specifies the multiple drill categories *objDrillCats*. *objDrillCats* is one or more drill categories, each identified by object name or object identifier.
Associations *assocopts*	Parameters that specify associations for the level. For the complete list of options, see assocopts.

Example

This example defines the level Year.

```
LevelMake 237 "Year" Blanks "(blank)" Inclusion Generate DateFunction Year
Generate Need RefreshLabel False RefreshDescription False RefreshShortName
False NewCatsLock False CatLabFormat "YYYY" Timerank 10 UniqueCategories True
UniqueMove False Associations 1251 "Time" AssociationType Type_Query
AssociationRole Role_Source AssociationReferenced "Time Associations
1253 "Time" AssociationContext 235 AssociationType Type_Query
```

```
AssociationRole Role_OrderBy AssociationReferenced "Time" SortOrder  Default
SortAs Ascending
```

LevelMoveAfter

The `LevelMoveAfter` verb switches the positions of two levels.

The Windows interface equivalent is to drag and drop the level into the required position.

Using this verb does not change the object identifier for the level or its position in the .mdl file. It only affects the visual display of the level on the Windows interface and its position in the .mdc file, which affects how it is viewed in the reporting components.

The syntax is as follows:

LevelMoveAfter *objLevel* [
Dimension *objDim*]
[**Drill** *objDrill*]
Child *objLevel*

Argument	Description
LevelMoveAfter *objLevel*	Specifies the level that is to be placed after. *objLevel* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the level. *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Is specified, if necessary, to uniquely identify the level. *objDrillCat* can be the object name, object identifier, or both.
Child *objLevel*	Specifies the level that is to be placed before. *objLevel* can be the object name or object identifier.

Example

This example moves the level Product Type after the level Product ID.

```
LevelMoveAfter "Product Type" Drill "By Product Line" Child "Product   ID"
```

LevelMoveBefore

The `LevelMoveBefore` verb switches the positions of two levels.

The Windows interface equivalent is to drag the level into the required position.

Using this verb does not change the object identifier for the level or its position in the .mdl file. It only affects the visual display of the level on the Windows interface and its position in the .mdc file, which affects how it is viewed in the reporting components.

The syntax is as follows:

```
LevelMoveBefore objLevel [
Dimension objDim]
[Drill objDrill]
Child objLevel
```

Argument	Description
LevelMoveBefore *objLevel*	Specifies the level that is to be placed before. *objLevel* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the level. *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Is specified, if necessary, to uniquely identify the level. *objDrillCat* can be the object name, object identifier, or both.
Child *objLevel*	Specifies the level that is to be placed after. *objLevel* can be the object name or object identifier. The child is mandatory.

Example

This example moves the level Product Line before the level Product ID.

```
LevelMoveBefore "Product Line" Drill "By Product Line" Child "Product ID"
```

LevelNewDrill

The `LevelNewDrill` verb creates a new drill path in a level.

The Windows interface equivalent is to drag levels and columns to form the new path.

The syntax is as follows:

```
LevelNewDrillobjLevel [
Dimension objDim]
[Drill objDrillCat]
 Drill objDrillCatChild objLevel
```

Argument	Description
LevelNewDrill *objLevel*	Specifies the level that is to have a new drill path. *objLevel* can be the object name, object identifier, or both.

Argument	Description
Dimension *objDim*	Is specified, if necessary, to uniquely identify the level. *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Specifies the old drill category. *objDrillCat* can be the object name, object identifier, or both.
Drill *objDrillCat*	Specifies the new drill category. *objDrillCat* can be the object name, object identifier, or both.
Child *objLevel*	Specifies the joining level. *objLevel* can be the object name or object identifier.

Example

This example adds a new drill path Territory in the level State with a joining level of City.

```
LevelNewDrill "State" Drill "Territory" Child "City"
```

LevelUpdate

The `LevelUpdate` verb updates an existing level.

The Windows interface equivalent is to modify the **Level** property sheet.

For more information about updating levels, see "LevelMake" (p. 157).

The syntax is as follows:

```
LevelUpdate objLevel [
Dimension objDim]
[Drill objDrillCat]
 [levelopts] [DrillList objDrillCatsEndList]
[Associations assocopts]
```

Argument	Description
LevelUpdate *objLevel*	Specifies the level to update. *objLevel* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the level. *objDim* can be the object name, object identifier, or both.
Drill *objDrillCat*	Is specified, if necessary, to uniquely identify the level. *objDrillCat* can be the object name, object identifier, or both.

Argument	Description
levelopts	Optional parameters that describe the level in greater detail. For the complete list of options, see levelopts. Previously set options are retained unless you change them with this command.
DrillList *objDrillCats*	Specifies the multiple drill categories *objDrillCats*. *objDrillCats* is one or more drill categories, each identified by object name or object identifier.
Associations *assocopts*	Optional parameters that specify associations for the level. For the complete list of options, see assocopts.

Example

This example changes the inclusion property of level Product Type to Suppress. Because the object name does not uniquely identify the level, the dimension and drill category are specified.

```
LevelUpdate "Product Type" Dimension "Products" Drill "By Product Line"
Inclusion Suppress
```

MDCClear

The `MDCClear` verb clears the processing status for the specified PowerCube. The default status is New.

There is no Windows interface equivalent, although the effects can be verified on the **Processing** tab of the **PowerCube** property sheet.

The syntax is as follows:

`MDCClear` *`objCube`*

Argument	Description
MDCClear *objCube*	Specifies the PowerCube that is to have its status flags cleared. *objCube* can be the object name, object identifier, or both.

Example

This example changes the status for the Great Outdoors Sales PowerCube from OK to New.

```
MDCClear "Great Outdoors Sales"
```

MeasureAdd

The `MeasureAdd` verb adds a measure to the model. If the measure object already exists, an error message is issued.

The Windows interface equivalent is to click the **Insert** option on the **Edit** menu when a measure is selected.

Every measure requires an `AllocationAdd` statement that contains the measure name and the option `TypeDefault`.

Transformer creates this statement automatically when a measure is added in the Windows interface or when you use the verb `ModelEnsureCompleteness`. Otherwise, you must create the statement. For more information, see "AllocationAdd" (p. 88).

For more information about creating measures, see "MeasureMake" (p. 164).

The syntax is as follows:

```
MeasureAdd objMeasure [
meaoptsAssociations]
 assocopts
```

Argument	Description
MeasureAdd *objMeasure*	Creates the measure *objMeasure*. *objMeasure* must be the object name and can include the object identifier.
meaopts	Optional parameters that describe the measure in greater detail. Some options are set by default if you do not set them. For the complete list of options, see meaopts.
Associations *assocopts*	Parameters that specify associations for the measure. For the complete list of options, see assocopts.

Example

This example adds the Staff Count measure from the Staff Count column.

```
MeasureAdd 863 "Staff Count" Missing N/A TimeStateRollup Average StorageFloat64
OutPutScale 0 Decimals 0 ReverseSign False IsCurrency False WeightId 899
DrillThrough False EndList Associations 1309 "Staff Count" AssociationType
Type_Query AssociationRole Role_Source AssociationReferenced "Staff Count"
```

MeasureDelete

The `MeasureDelete` verb removes a measure from the model.

The Windows interface equivalent is to click **Delete** on the **Edit** menu when a measure is selected.

The syntax is as follows:

MeasureDelete *objMeasure*

Argument	Description
MeasureDelete *objMeasure*	Deletes the category *objMeasure*. *objMeasure* can be the object name, object identifier, or both.

Example

This example deletes the measure Profit Margin %.

```
MeasureDelete "Profit Margin %"
```

MeasureListUpdate

The `MeasureListUpdate` verb reorders the items in the list of measures.

The Windows interface equivalent is to drag the measures into the required order.

The syntax is as follows:

MeasureListUpdate *objMeasures* **EndList**

Argument	Description
MeasureListUpdate *objMeasures*	Lists the measures in the desired order. If not all measures in the model are listed, those listed are moved to the top of the list. *objMeasures* can be the object names or object identifiers.

Example

This example changes the order of the list of measures, placing Product Cost and then Revenue at the top.

```
MeasureListUpdate "Product Cost" "Revenue" EndList
```

MeasureMake

The `MeasureMake` verb creates a measure or updates an existing one.

The Windows interface equivalent, if the measure exists, is to modify the **Measure** property sheet. For a new measure, click the **Insert** option on the **Edit** menu when a measure is selected.

Every measure requires an `AllocationAdd` statement that contains the measure name and the option `Type Default`. Transformer creates this statement automatically when a measure is added on the Windows interface or when you use the verb `ModelEnsureCompleteness`. Otherwise, you must create the statement. For more information, see "AllocationAdd" (p. 88).

Measure names cannot contain an at sign (`@`).

The syntax is as follows:

MeasureMake *objMeasure* [
meaopts] **Associations** *assocopts*

Argument	Description
MeasureMake *objMeasure*	Creates the measure *objMeasure* or modifies it if it exists. *objMeasure* can be the object name, object identifier, or both. Include the object name if the category does not exist.
meaopts	Optional parameters that describe the measure in greater detail. Some options are set by default if you do not set them. For the complete list of options, see meaopts. If the category exists, previously set options are retained unless you change them with this command.
Associations *assocopts*	Parameters that specify associations for the measure. For the complete list of options, see assocopts.

Example

The following statements were generated when a sample model created on the Windows interface was saved as an .mdl file.

This example creates or updates the StaffCount measure.

```
MeasureMake 863 "Staff Count" Missing N/A TimeStateRollup Average Storage
Float64 OutPutScale 0 Decimals 0 ReverseSign False IsCurrency False WeightId 899
DrillThrough False EndList Associations 1309 "Staff
Count" AssociationType Type_Query AssociationRole Role_
Source AssociationReferenced "Staff Count"
```

This example creates or updates the calculated measure Profit Margin %.

```
MeasureMake "Profit Margin %" Calc ("Revenue@259" - "Product Cost@261") /
"Revenue@259" Missing N/A Timing After_Rollup Storage Float64 Scale 0 Decimals
 1 Sign False Format "0%~1" MeasureInfo "Percent profit" DrillThrough False
EndList
```

This example creates or updates a Revenue measure that allows drill-through to an IBM Cognos Impromptu report (dt_cust.imr).

```
MeasureMake "Revenue" Association "Revenue" Storage Default Scale 0 Decimals
0 Sign False Format "#,##0~0" MeasureInfo "Gross revenue from product sales.
" DrillThrough True "c:\dt_cust.imr" "" EndList
```

MeasureUpdate

The `MeasureUpdate` verb updates an existing measure.

The Windows interface equivalent is to modify the **Measure** property sheet.

For more information about updating measures, see "MeasureMake" (p. 164).

The syntax is as follows:

```
MeasureUpdate objMeasure [
meaopts] [Associations assocopts]
```

Argument	Description
MeasureUpdate *objMeasure*	Specifies the measure to update. *objMeasure* can be the object name, object identifier, or both.
meaopts	Optional parameters that describe the measure in greater detail. For the complete list of options, see meaopts. If the category exists, previously set options are retained unless you change them with this command.
Associations *assocopts*	Optional parameters that specify associations for the measure. For the complete list of options, see assocopts.

Example

This example changes the timing of the Profit Margin % measure.

```
MeasureUpdate "Profit Margin %" Timing Before_Rollup
```

ModelEnsureCompleteness

The `ModelEnsureCompleteness` verb scans all dimensions and cubes in the model and creates necessary default settings and objects, including root categories, drill categories, and default dimension views. There is no Windows interface equivalent, as this is done automatically by the Transformer component.

Using `ModelEnsureCompleteness` avoids the need to create Dimension views and associate them with PowerCubes and security objects in the model. We recommend that you include this verb near the end of your user-defined models, to ensure that each .mdl file is valid.

The syntax is as follows:

```
ModelEnsureCompleteness
```

NewModel

The `NewModel` verb creates, names, describes, and opens a Transformer model.

The Windows interface equivalent is to click the **New** option on the **File** menu.

Transformer saves the model file in the My Documents/Transformer/Models directory, unless a server path is specified.

Running `NewModel` will close previously opened files without saving them. If you want to save a file, you must use the verb `SaveMDL` or `SavePY` before you use the `NewModel` verb.

The syntax is as follows:

`NewModel` *`objModel`* [
`appqueryopts`]

Argument	Description
NewModel *objModel*	Specifies the model that is to be opened. *objModel* must be the model name.
appqueryopts	Optional parameters that describe the model in greater detail. Some options are set by default if you do not set them. For the complete list of options, see appqueryopts.

Example

This example creates a model called PowerPlay Sample, and provides a description.

```
NewModel "PowerPlay Sample" AppInfo "This model used the tutorial."
```

OpenDef

Opens .def, .gen, and .dat files.

The Windows interface equivalent is the **Open** dialog box, in editions of Transformer where you can select **PowerPlay definition files (*.def & *.gen)**.

Support for this format was retained to ensure compatibility with earlier versions.

The syntax is as follows:

`OpenDef` *`filename`*

Argument	Description
OpenDef *filename*	Specifies the file name, including path if desired.

Example

This example opens the "file.def" PowerPlay definition file.

```
OpenDef "file.def"
```

OpenMDL

The `OpenMDL` verb opens the specified .mdl file.

The Windows interface equivalent is the **Open** dialog box, where you can select **Model files (*.py? and *.mdl)**.

Execution of the remaining MDL script continues after the file is opened.

You can only open one model at a time. Any model that was open before the `OpenMDL` command is issued will be closed without being saved.

To save an open model, you must use the `SaveMDL` or `SavePY` command before using the `OpenMDL` command.

The syntax is as follows:

`OpenMDL` *`filename`*

Argument	Description
OpenMDL *filename*	Specifies the file name, including path if desired, for the current model. The file name should have the extension .mdl.

Example

This example opens the model outdoors.mdl, located in the root directory of the c: drive.

```
OpenMDL "c:\outdoors.mdl"
```

OpenPY

The `OpenPY` verb opens the specified .py? file. The question mark in the extension .py? is replaced by the character that is used in your release of Transformer, such as .pyj.

The Windows interface equivalent is the **Open** dialog box, where you can select **Model files (*.py? and *.mdl)**.

Execution of the remaining MDL script continues after the file is opened.

You can only open one model at a time. Any model that was open before the `OpenPY` command is issued will be closed without being saved.

To save an open model, you must use the `SaveMDL` or `SavePY` command before the `OpenMPY` command.

The syntax is as follows:

`OpenPY` *`filename`*

Argument	Description
OpenPY *filename*	Specifies the file name, including path if desired, for the current model. The file name should have the extension .py?, where ? is dependent on your version of Transformer.

Example

This example opens Model.pyj in the root directory of the c: drive and saves it as an .mdl file.

```
OpenPY "C:\Model.pyj" SaveMDL "C:\Model.mdl"
```

PopulateFromQueries

The `PopulateFromQueries` verb populates the model with only those categories found in the specified data sources.

The Windows interface equivalent is to right-click one or more data sources and, from the **Run** menu, click **Generate Categories**.

The syntax is as follows:

`PopulateFromQueries` *`objQueries`*

Argument	Description
objQueries	Specifies one or more data sources to be used. *objQueries* can be the object identifiers or object names.

Example

This example populates the model using only the All Staff Count (Excel) data source.

```
PopulateFromQueries "All Staff Count (Excel)"
```

PopulateModel

The `PopulateModel` verb populates the model with categories using all data sources in the model.

The Windows interface equivalent is the **Generate Categories** option on the **Run** menu.

In IBM Cognos 8, you do not specify the platform type.

The syntax is as follows:

`PopulateModel`

PowerCubeCustomViewListUpdate

The `PowerCubeCustomViewListUpdate` verb defines the list of custom views assigned to a cube.

The syntax is as follows:

`PowerCubeCustomViewListUpdate` *`PowerCube ID`* **`StartList`** *`PowerCube ID`* **`Endlist`**

Example

```
PowerCubeCustomViewChildListUpdate 11535 StartList 11537
Endlist
```

PowerCubeDelete

The `PowerCubeDelete` verb removes the specified PowerCube or cube group by deleting the PowerCube objects from the model. This verb does not affect existing .mdc files.

The Windows interface equivalent is to click **Delete** on the **Edit** menu when a PowerCube or cube group is selected.

The syntax is as follows:

```
PowerCubeDelete {objCube|
objCubeGroup}
```

Argument	Description
PowerCubeDelete {*objCube*|*objCubeGroup*}	Deletes the PowerCube *objCube* or the PowerCube group *objCubeGroup*. *objCube* and *objCubeGroup* can be the object name, object identifier, or both.

Example

This example deletes the PowerCube **Great Outdoors Sales**.

```
PowerCubeDelete "Great Outdoors Sales"
```

PowerCubeListUpdate

The `PowerCubeListUpdate` verb reorders the PowerCubes and cube groups listed in the model.

The Windows interface equivalent is to drag PowerCubes and cube groups into the required order.

The syntax is as follows:

```
PowerCubeListUpdate [objCubes]
[objCubeGroups] EndList
```

Argument	Description
objCubes	Lists the PowerCubes in the desired order. If not all cubes are specified, those specified move to the top of the list. The cubes can be identified by object name or object identifier.
objCubeGroups	Lists the cube groups in the desired order. If not all groups are specified, those specified move to the top of the list. The cube groups can be identified by object name or object identifier.

Example

This example changes the order of the Great Outdoors Sales and cubegroup1 cubes.

```
PowerCubeListUpdate "cubegroup1" "Great Outdoors Sales" EndList
```

PromptAdd

The `PromptAdd` verb imports prompts defined in IBM Cognos 8 package and report data sources.

PromptAdd *promptname* **DataSource** *datasource ID promptopts*

Argument	Description
prompt name	Specifies the name of the prompt.
data source name	Specifies the name of the IBM Cognos 8 package, report, or query data source.
promptopts	Specifies the prompt type and prompt value. Simple-value, multi-value, and range prompts are type "xsdString"; Member Unique Name (MUN) prompts, for SAP and DMR data sources only, are type "memberUniqueName".

Example

This example adds a prompt named Product Name 1 with type xsdString and value BugShield Natural for the package 574942~1.

```
PromptAdd "Product Name1" DataSource "574942~1" PromptType
"xsdString"
PromptValue "BugShield Natural"
```

PromptDelete

The `PromptDelete` verb deletes prompts defined in Transformer model.

PromptDelete *promptname*

Argument	Description
prompt name	Specifies the name of the prompt.

Example

This example deletes the Product Name1 prompt from the model.

```
PromptDelete "Product Name1"
```

PromptMake

The `PromptMake` verb creates prompts that are defined in IBM Cognos 8 package and report data sources.

PromptMake *prompt name* **DataSource** *data source ID promptopts*

Argument	Description
prompt name	Specifies the name of the prompt.
data source name	Specifies the name of the IBM Cognos 8 package, report, or query data source.
promptopts	Specifies the prompt type and prompt value. Simple-value, multi-value, and range prompts are type "xsdString"; Member Unique Name (MUN) prompts, for SAP and DMR data sources only, are type "memberUniqueName".

Example

This example creates a prompt named Product Name 1 with type xsdString and value BugShield Natural for the package 574942~1.

```
PromptMake "Product Name1" DataSource "574942~1" PromptType
"xsdString"
PromptValue "BugShield Natural"
```

PromptUpdate

The `PromptUpdate` verb updates prompts in the Transformer model.

```
PromptUpdate prompt
name DataSource data
source ID promptopts
```

Argument	Description
prompt name	Specifies the name of the prompt.
data source name	Specifies the name of the IBM Cognos 8 package, report, or query data source.
promptopts	Specifies the prompt type and prompt value. Simple-value, multi-value, and range prompts are type "xsdString"; Member Unique Name (MUN) prompts, for SAP and DMR data sources only, are type "memberUniqueName".

Example

This example updates a prompt named Product Name 1 with type xsdString and value SmallShield Artificial for the package 574942~1.

```
PromptUpdate "Product Name1" DataSource "574942~1" PromptType
"xsdString"
PromptValue "SmallShield Artificial"
```

ReportPartitions

The `ReportPartitions` verb writes a partition status report to the log file for a specified PowerCube. There is no Windows interface equivalent.

The syntax is as follows:

`ReportPartitions` *`objCube`*

Argument	Description
ReportPartitions *objCube*	Specifies the PowerCube that is to have a partition status report written to its log file.

Example

This example writes a partition report to the log file for the Great Outdoors Sales PowerCube.

```
ReportPartitions "Great Outdoors Sales"
```

RootCatMake

The `RootCatMake` verb creates a root category or updates an existing one.

The Windows interface equivalent is to modify the **Root Category** property sheet. Root categories are created automatically when modeling on the Windows interface.

The syntax is as follows:

`RootCatMake` *`objRootCat`* `[`
`Dimension` *`objDim`*`]`
`[`*`catopts`*`]`

Argument	Description
RootCatMake *objRootCat*	Creates the category *objRootCat* or changes it if it exists. *objRootCat* can be the object name, object identifier, or both. Include the object name if the category does not exist.
Dimension *objDim*	Specifies the dimension of a new root category or, if necessary, uniquely identifies an existing root category. *objDim* can be the object name, object identifier, or both.
catopts	Optional parameters that describe the category in greater detail. Some options are set by default if you do not set them. For the complete list of options, see catopts. If the category exists, previously set options are retained unless you change them with this command.

Example

The example defines the root category Product Line.

```
RootCatMake "Product Line" Dimension "Products" Inclusion Generate Lastuse
19971202 Filtered False Suppressed False Sign False IsKeyOrphanage
False IsTruncated False Blanks False
```

RootCatUpdate

The `RootCatUpdate` verb updates an existing root category.

The Windows interface equivalent is to modify the **Root Category** property sheet.

For more information about updating root categories, see "RootCatMake" (p. 173).

The syntax is as follows:

RootCatUpdate *objRootCat* **Dimension** [*objDim*]
[*catopts*]

Argument	Description
RootCatUpdate *objRootCat*	Specifies the root category to update. *objRootCat* can be the object name, object identifier, or both
Dimension *objDim*	Required. Is specified to uniquely identify the root category. *objDim* can be the object name, object identifier, or both.
catopts	Optional parameters that describe the category in greater detail. For the complete list of options, see catopts. Previously set options are retained unless you change them with this command.

Example

This example adds the description "Products carried by the Great Outdoors Company" to the Products root category.

```
RootCatUpdate "Products" Dimension "Products" Catinfo "Products carried by
the Great Outdoors Company."
```

SaveMDL

The `SaveMDL` verb saves the current model as an .mdl file.

The Windows interface equivalent is to save the model with an .mdl extension.

The syntax is as follows:

SaveMDL *filename*

Argument	Description
SaveMDL *filename*	Specifies the file name, including path if desired, for the current model. The file name should have the extension .mdl.

Example

This example opens a .py? file and saves it as an .mdl file.

```
OpenPY "C:\model.pyj" SaveMDL "C:\model.mdl"
```

SavePY

The `SavePY` verb saves the current model as a .py? file, where the ? is replaced by the character that is used in your release of Transformer.

The Windows interface equivalent is to save the model with a .py? extension.

The syntax is as follows:

SavePY *filename*

Argument	Description
SavePY *filename*	Specifies the file name, including path if desired, for the current model. The file name should have the extension .py?, where ? is replaced by the character that is used in your release of Transformer.

Example

This example saves the open model in .py format in the root directory of c: drive, and calls it model.pyj.

```
SavePY "C:\model.pyj"
```

SecurityNamespaceAdd

The `SecurityNamespaceAdd` verb defines the IBM Cognos 8 namespace from which the IBM Cognos 8 security objects (groups, roles, users) are retrieved.

The Windows interface equivalent is to click **Assign Security** in the **Custom View** dialog box.

The syntax is as follows:

SecurityNameSpaceAdd *"namespacename"* **SecurityNamespaceCAMID** *'CAMID(":")'*

Argument	Description
SecurityNameSpaceAdd "*namespace name*"	Specifies the namespace from which to retrieve the IBM Cognos 8 security objects. The namespace name is defined in IBM Cognos Configuration.
SecurityNamespaceCAMID '*CAMID(":")*'	Defines the search path for the CAMID, or security object ID.

Example

This example adds the namespace "authors" to the membership in the model.

```
SecurityNameSpaceAdd "authors" SecurityNamespaceCAMID
'CAMID(":")'
```

SecurityNamespaceDelete

The `SecurityNamespaceDelete` verb deletes the IBM Cognos 8 namespace from the model.

The Windows interface equivalent is to click **Unassign Security** in the **Custom View** dialog box.

The syntax is as follows:

`SecurityNameSpaceDelete` *"namespace name"*

Argument	Description
SecurityNameSpaceDelete "*namespace name*"	Specifies the namespace to be deleted from the model. The namespace name is defined in IBM Cognos Configuration.

Example

This example deletes the namespace "authors" from the membership in the model.

```
SecurityNameSpaceDelete "authors"
```

SecurityNamespaceMake

The `SecurityNamespaceMake` verb defines the IBM Cognos 8 namespace from which the IBM Cognos 8 security objects (groups, roles, users) are retrieved.

The Windows interface equivalent is to click **Assign Security** in the **Custom View** dialog box.

The syntax is as follows:

`SecurityNameSpaceMake` *"namespacename"* **`SecurityNamespaceCAMID`** *'CAMID(":")'*

Argument	Description
SecurityNameSpaceMake "*namespace name*"	Specifies the namespace from which to retrieve the IBM Cognos 8 security objects. The namespace name is defined in IBM Cognos Configuration.
SecurityNamespaceCAMID '*CAMID(":")*'	Defines the search path for the CAMID, or security object ID.

Example

This example adds the namespace "authors" to the membership in the model.

```
SecurityNameSpaceMake "authors" SecurityNamespaceCAMID
'CAMID(":")'
```

SecurityNamespaceUpdate

The `SecurityNamespaceUpdate` verb updates the IBM Cognos 8 namespace from which the IBM Cognos 8 security objects (groups, roles, users) are retrieved.

The Windows interface equivalent is to modify the assigned security membership in the **Custom View** dialog box.

The syntax is as follows:

SecurityNameSpaceUpdate *"namespace name"* **SecurityNamespaceCAMID** *'CAMID(":")'*

Argument	Description
SecurityNameSpaceUpdate "*namespace name*"	Specifies the namespace from which to retrieve the IBM Cognos 8 security objects. The namespace name is defined in IBM Cognos Configuration.
SecurityNamespaceCAMID '*CAMID(":")*'	Defines the search path for the CAMID, or security object ID.

Example

This example updates the IBM Cognos 8 authors namespace.

```
SecurityNameSpaceUpdate "authors" SecurityNamespaceCAMID
'CAMID(":")'
```

SecurityObjectAdd

The `SecurityObjectAdd` verb defines a security object to be imported into the model from the last defined namespace.

The syntax is as follows:

```
SecurityObjectAdd securityobject
IDsecurity object name SecurityNamespace namespace SecurityObjectDisplayName
securityobject name SecurityObjectType SecurityType_Group|SecurityType_Role|
SecurityType_User
CustomViewList custom viewlist
```

Argument	Description
SecurityObjectAdd *security object ID security object name*	Specifies the security object to be imported into the model from the last defined namespace.
SecurityNamespace *namespace*	Specifies the namespace where the security object exists. References can be the ID or the namespace name.
SecurityObjectDisplayName *security object name*	Specifies the security object name that is displayed in the model.
SecurityObjectType SecurityType_Group|SecurityType_Role|SecurityType_User	Specifies the security object type. Type can be a group, role, or user.
CustomViewList *custom view list*	Specifies the custom views to which the security object is assigned.

Example

The following example adds the security role "Authors"(ID 11531) from the namespace 11533 and assigns it to custom view 11529.

```
SecurityObjectAdd 11531 'CAMID (":Authors")' SecurityNamespace
11533
SecurityObjectDisplayName "Authors" SecurityObjectType SecurityType_Role
CustomViewList
11529 EndList
```

SecurityObjectDelete

The `SecurityObjectDelete` verb removes a security object from the model.

The syntax is as follows:

```
SecurityObjectDelete securityobject
IDsecurity object name
```

Argument	Description
SecurityObjectDelete *security object ID security object name*	Specifies the security object to be removed from the model.

Example

The following example removes the security role "Authors"(ID 11531) from the model.

```
SecurityObjectDelete 11531 'CAMID (":Authors")'
```

SecurityObjectMake

The `SecurityObjectMake` verb defines a security object to be imported into the model from the last defined namespace.

The syntax is as follows:

```
SecurityObjectMake securityobject
IDsecurity object name SecurityNamespace namespace SecurityObjectDisplayName
securityobject name SecurityObjectType SecurityType_Group|SecurityType_Role|
SecurityType_User
CustomViewList custom viewlist
```

Argument	Description
SecurityObjectMake *security object ID security object name*	Specifies the security object to be imported into the model from the last defined namespace.
SecurityNamespace *namespace*	Specifies the namespace where the security object exists. References can be the ID or the namespace name.
SecurityObjectDisplayName*security object name*	Specifies the security object name that is displayed in the model.
SecurityObjectType SecurityType_Group|SecurityType_Role|SecurityType_User	Specifies the security object type. Type can be a group, role, or user.
CustomViewList *custom view list*	Specifies the custom views to which the security object is assigned.

Example

The following example imports the security role "Authors"(ID 11531) from the namespace 11533 and assigns it to custom view 11529.

```
SecurityObjectMake 11531 'CAMID (":Authors")' SecurityNamespace
11533
SecurityObjectDisplayName "Authors" SecurityObjectType SecurityType_Role
CustomViewList
11529 EndList
```

SecurityObjectUpdate

The `SecurityObjectUpdate` verb updates a security object in the model.

The syntax is as follows:

```
SecurityObjectUpdate securityobject
IDsecurity object name SecurityNamespace namespace SecurityObjectDisplayName
securityobject name SecurityObjectType SecurityType_Group|SecurityType_Role|
SecurityType_User
CustomViewList custom viewlist
```

Argument	Description
SecurityObjectUpdate *security object ID security object name*	Specifies the security object in the model to be updated.
SecurityNamespace *namespace*	Specifies the namespace where the security object exists. References can be the ID or the namespace name.
SecurityObjectDisplayName*security object name*	Specifies the security object name that is displayed in the model.
SecurityObjectType SecurityType_Group|SecurityType_Role|SecurityType_User	Specifies the security object type. Type can be a group, role, or user.
CustomViewList *custom view list*	Specifies the custom views to which the security object is assigned.

Example

The following example updates the security role "Authors"(ID 11531) from the namespace 11533 by assigning it to an additional custom view 11539).

```
SecurityObjectUpdate 11531 'CAMID (":Authors")' SecurityNamespace
11533
SecurityObjectDisplayName "Authors" SecurityObjectType SecurityType_Role
CustomViewList
11529 11539 EndList
```

SignonAdd

The `SignonAdd` verb is used to add IBM Cognos 8 or data source signon objects in the Transformer model.

The Windows interface equivalent is to click the **Insert Signon** option on the **Edit** menu, and follow the prompts to specify the **Signon** properties.

There is no order requirement to the syntax for SignonAdd.

The syntax for a Cognos 8 signon is as follows:

```
SignonAdd "signonname" UserId "user name" password"password" AutoLogon
{True|False} SignonNamespace "namespace name" SignonType"Cognos"
```

The syntax for a data source signon is as follows:

```
SignonAdd "signonname" PromptForPassword {False|True} UserId "user name" password
 "password" SignonType"
DataSource"
```

Argument	Description
SignonAdd *objSignon*	Adds the signon *objSignon*. *objSignon* must be the object name, and can include the object identifier.
signonopts	Optional parameters that describe the signon object in greater detail. Some options are set by default if you do not set them. For the complete list of options, see signonopts.

Example

This example creates two Cognos 8 signons and two data source signons:

- LDAP1 is a Cognos 8 signon with AutoLogon enabled.
- LDAP2 is a Cognos 8 signon with AutoLogon disabled.
- DataSource1 is a data source signon that does not prompt the user to enter a password.
- DataSource2 is a data source signon that prompts the user to enter a password.

```
SignonAdd "LDAP1" UserId "user1" password "pwd1" AutoLogon True SignonNamespace "LDAP" SignonType "Cognos"

SignonAdd "LDAP2" UserId "user2" password "pwd2" AutoLogon False SignonNamespace "LDAP" SignonType "Cognos"

SignonAdd "DataSource1" PromptForPassword False UserId "gosales" password "gosales" SignonType "DataSource"

SignonAdd "DataSource2" PromptForPassword True UserId "gosales" password "gosales" SignonType "DataSource"
```

SignonDelete

The `SignonDelete` verb deletes a Cognos 8 or data source signon from the model.

The Windows interface equivalent is to click **Delete** on the **Edit** menu when an object in the **Signon List** is selected.

The syntax is as follows:

```
SignonDelete objSignon
```

Argument	Description
SignonDelete *objSignon*	Specifies the signon that is to be deleted. *objSignon* can be the object name, object identifier, or both.

Example

This example deletes the signon I40_DBASE_NTV_PP74_SAMPLE.

```
SignonDelete "I40_DBASE_NTV_PP74_SAMPLE"
```

SignonMake

The `SignonMake` verb is used to create Cognos 8 or data source signon objects.

The Windows interface equivalent is to click the **Insert Signon** option on the **Edit** menu, and follow the prompts to specify the **Signon** properties.

There is no order requirement to the syntax for SignonMake.

The syntax for a Cognos 8 signon is as follows:

```
SignonMake "signonname" UserId "user name" password"password" AutoLogon
{True|False} SignonNamespace"namespace name" SignonType "Cognos"
```

The syntax for a data source signon is as follows:

```
SignonMake "signonname" PromptForPassword {False|True} UserId "user name"
password "password" SignonType
"DataSource"
```

Argument	Description
SignonMake *objSignon*	Creates the signon *objSignon*. *objSignon* must be the object name, and can include the object identifier.
signonopts	Optional parameters that describe the signon object in greater detail. Some options are set by default if you do not set them. For the complete list of options, see signonopts.

Example

This example creates two Cognos 8 signons and two data source signons:

- LDAP1 is a Cognos 8 signon with **AutoLogon** enabled.
- LDAP2 is a Cognos 8 signon with **AutoLogon** disabled.
- DataSource1 is a data source signon that does not prompt the user to enter a password.
- DataSource2 is a data source signon that prompts the user to enter a password.

SignonMake "LDAP1" **UserId** "user1" **password** "pwd1" **AutoLogon** True **SignonNamespace** "LDAP" **SignonType** "Cognos"

SignonMake "LDAP2" **UserId** "user2" **password** "pwd2" **AutoLogon** False **SignonNamespace** "LDAP" **SignonType** "Cognos"

SignonMake "DataSource1" **PromptForPassword** False **UserId** "gosales" **password** "gosales" **SignonType** "DataSource"

```
SignonMake "DataSource2" PromptForPassword True UserId "gosales" password "gosales"
SignonType "DataSource"
```

SignonUpdate

The `SignonUpdate` verb updates Cognos 8 or data source signon objects in the model.

The Windows interface equivalent to modify the **Signon** property sheet.

The syntax for a Cognos 8 signon is as follows:

```
SignonUpdate "signonname" UserId "user name" password "password" AutoLogon
{True|False} SignonNamespace "namespace name" SignonType "Cognos"
```

The syntax for a data source signon is as follows:

```
SignonUpdate "signonname" PromptForPassword {False|True} UserId "username"
password "password" SignonType
"DataSource"
```

Argument	Description
SignonListUpdate *objSignons*	Lists the signon objects in the desired order. If not all signons are specified, the specified ones are moved to the top of the list. *objSignons* can be the object names or object identifiers.
signonopts	Optional parameters that describe the database connection in greater detail. For the complete list of options, see signonopts. Previously set options are retained unless you change them with this command.

Example

This example updates the LDAP3 Cognos 8 signon object.

```
SignonUpdate "LDAP3" UserId "user3" password "pwd4" AutoLogon
False SignonNamespace "LDAP" SignonType "Cognos"
```

SourceListUpdate

The `SourceListUpdate` verb reorders the items in the list of data sources for the model.

The Windows interface equivalent is to drag the data source objects into the required position in the list.

The syntax is as follows:

```
SourceListUpdate objDataSources EndList
```

Argument	Description
SourceListUpdate *objDataSources*	Lists the data sources in the desired order. If not all data sources are specified, the specified ones are moved to the top of the list. The data sources can be identified by object name or object identifier.

Example

This example reorders the list of data sources so that Locations, Products, and Main are the first three in the list.

```
SourceListUpdate "Locations (CSV)" "Products (CSV)" "MAIN (IQD)" EndList
```

SpecialCatAdd

The `SpecialCatAdd` verb creates a special category in the specified dimension.

The Windows interface equivalent is to open the **Diagram**, click the right side of a root or special category, and drag the connection to the right.

For more information about creating special categories, see "SpecialCatMake" (p. 185).

The syntax is as follows:

```
SpecialCatAdd objSpecialCat [
Dimension objDim]
Parent objCat [
catopts]
```

Argument	Description
SpecialCatAdd *objSpecialCat*	Creates the special category *objSpecialCat*. *objSpecialCat* must be the object name and can include the object identifier.
Dimension *objDim*	Specifies a dimension *objDim* for the special category. *objDim* can be the object name, object identifier, or both.
Parent *objCat*	Specifies the parent category *objCat*. *objCat* can be the object name or object identifier.
catopts	Optional parameters that describe the category in greater detail. Some options are set by default if you do not set them. For the complete list of options, see catopts.

Example

This example creates the special category Current Month.

```
SpecialCatAdd "Current Month" Parent "Time"
```

SpecialCatDelete

The `SpecialCatDelete` verb removes a special category from the model.

The Windows interface equivalent is to click **Delete** on the **Edit** menu when a special category is selected.

If a special category has children, the children must be deleted before the special category is deleted or else an error message is issued. This is different from the Windows interface, where deleting a special category automatically deletes its children.

The syntax is as follows:

```
SpecialCatDelete objSpecialCat [
Dimension objDim]
```

Argument	Description
SpecialCatDelete *objSpecialCat*	Deletes the special category *objSpecialCat*. *objSpecialCat* can be the object name, object identifier, or both.
Dimension *objDim*	Is specified, if necessary, to uniquely identify the category. *objDim* can be the object name, object identifier, or both.

Example

This example deletes the special category Current Month.

```
SpecialCatDelete "Current Month"
```

SpecialCatMake

The `SpecialCatMake` verb creates a special category or updates an existing one.

The Windows interface equivalent, if the special category exists, is to modify the **Special Category** property sheet. For a new special category, open the **Diagram**, click the right side of the root or special category, and drag the connection to the right.

Calculated categories are defined as regular categories, not special categories, in MDL. For more information, see "DimCalcDefMake" (p. 148).

Note: Category object names differ between MDL and the Windows interface. MDL uses the category code as the object name. For more information, see "Locating Objects Uniquely" (p. 24).

The syntax is as follows:

```
SpecialCatMake objSpecialCat [
Dimension objDim]
```

```
[Parent objLevel]
[catopts]
```

Argument	Description
SpecialCatMake *objSpecialCat*	Creates the category *objSpecialCat* or modifies it if it exists. *objSpecialCat* can be the object name, object identifier, or both. Include the object name if the special category does not exist.
Dimension *objDim*	When creating a special category, this specifies the dimension in which the category is to be placed. When updating a category, this is used, if necessary, to uniquely identify the category. *objDim* can be the object name, object identifier, or both.
Parent *objLevel*	When creating a category, this specifies its parent and is mandatory. This argument is not needed when updating a category. *objLevel* can be object name or object identifier.
catopts	Optional parameters that describe the category in greater detail. Some options are set by default if you do not set them. For the complete list of options, see catopts. If the category exists, previously set options are retained unless you change them with this command.

Example

This example defines the special category `Current Month`.

```
SpecialCatMake "Current Month" Parent "Time" Lastuse 19970425 Rollup
True TimeAggregate Single RunningPeriods 0 TargetOffset 0 TargetLevel
"Month" ContextOffset 0 DateDrill 5237 SplitWeek False Primary 5897
Filtered False Suppressed False Sign False IsKeyOrphanage False IsTruncated
False Blanks False
```

SpecialCatUpdate

The `SpecialCatUpdate` verb updates an existing special category.

The Windows interface equivalent is to modify the **Special Category** property sheet.

For more information about updating special categories, see "SpecialCatMake" (p. 185).

The syntax is as follows:

```
SpecialCatUpdate objSpecialCat [
Dimension objDim]
[catopts]
```

Argument	Description
SpecialCatUpdate*objSpecialCat*	Updates the special category *objSpecialCat*. *objSpecialCat* can be the object name, object identifier, or both.
Dimension*objDim*	Is specified, if necessary, to uniquely identify the special category. *objDim* can be the object name, object identifier, or both.
catopts	Optional parameters that describe the category in greater detail. For the complete list of options, see catopts. Previously set options are retained unless you change them with this command.

Example

This example turns off the RollUp attribute for the special category Current Month.

```
SpecialCatUpdate "Current Month" RollUp False
```

SubDimRootMake

The `SubDimRootMake` verb creates a subdimension or updates the category at the root of an existing subdimension.

The Windows interface equivalent, if the subdimension exists, is to modify the **Category** property sheet of the category that is at the root of the subdimension. For a new dimension, click the **Create/Delete Subdimension** option on the **Diagram** menu when a category is selected.

The category acting as the root category of the subdimension is a regular category, and has the attributes of a regular category.

If you create a subdimension in the Windows interface and save the model as an .mdl file, when the MDL is generated for the subdimension, the following changes occur:

- A new `SubDimRootMake` statement replaces the `CatMake` statement for the category.
- New `DrillCatMake` and `LevelMake` statements are created and placed immediately after the `SubDimRootMake` statement.
- All of the categories in the subdimension remain but have new parents, drills, and levels.

Note: Category object names differ between MDL and the Windows interface. MDL uses the category code as the object name. For more information, see "Locating Objects Uniquely" (p. 24).

The syntax is as follows:

```
SubDimRootMake objCat [
Dimension objDim]
[Parent objLevel]
[Drill objDrillCat]
[Levels objLevel]
[catopts]
```

Argument	Description
SubDimRootMake *objCat*	Specifies the category that is to be at the root of the subdimension. *objCat* can be the object name, object identifier, or both.
Dimension *objDim*	When creating a subdimension or updating the root category of a subdimension, this argument is used if necessary, to uniquely identify it. *objDim* can be the object name, object identifier, or both.
Parent *objLevel*	When creating a subdimension, this specifies the parent and must be included. *objLevel* can be object name or object identifier.
Drill *objDrillCat*	When creating a subdimension, this specifies the drill category and is optional. *objDrillCat* can be the drill category object name, object identifier, or both.
Levels *objLevel*	When creating a subdimension, this specifies its level and must be included. *objLevel* can be the object name or object identifier.
catopts	Optional parameters that describe *objCat* in greater detail. Some options are set by default if you do not set them. For the complete list of options, see catopts. If the category exists, previously set options are retained unless you change them with this command.

Example

This example defines the root category 2006 for the subdimension Years.

```
SubDimRootMake "2006" Dimension "Years" Parent "By Time" Drill "By Time"
Levels "Year" Label "2006"
```

SubDimRootUpdate

The `SubDimRootUpdate` verb updates the category at the root of an existing subdimension.

The Windows interface equivalent is to modify the property sheet of the category that is at the root of the subdimension.

For more information about subdimensions, see "SubDimRootMake" (p. 187).

The syntax is as follows:

```
SubDimRootUpdate objCatDimension objDim [
catopts]
```

Argument	Description
SubDimRootUpdate *objCat*	Updates the category that is at the root of the subdimension. *objCat* can be the object name, object identifier, or both.
Dimension *objDim*	Specifies the dimension in which the subdimension is located. *objDim* can be the object name, object identifier, or both. This argument is mandatory.
catopts	Optional parameters that describe the category in greater detail. For the complete list of options, see catopts. Previously set options are retained unless you change them with this command.

Example

This example modifies the category Sport Wear, which is at the root of a subdimension, by adding the description "Sports clothing".

```
SubDimRootUpdate "Sport Wear" Dimension "Products" Parent "GO Sport Line"
Drill "By Product Line" Levels "Product Type" CatInfo "Sports clothing"
```

SummarizeCat

The `SummarizeCat` verb summarizes the data for the descendants of a specified category in any PowerCube that is created using the specified dimension view.

The Windows interface equivalent is to click the **Summarize** option on the **Diagram** menu when a category is selected.

Using `SummarizeCat` on an already-excluded category removes the filter.

Note: Category object names differ between MDL and the Windows interface. MDL uses the category code as the object name. For more information, see "Locating Objects Uniquely".

The syntax is as follows:

```
SummarizeCat objView [
Dimension objDim]
Category objCat
```

Argument	Description
SummarizeCat *objView*	Specifies the dimension view. *objView* can be the object name, object identifier, or both.
Dimension *objDim*	Must be specified if the view is not referenced by object identifier. *objDim* can be the object name, object identifier, or both.
Category *objCat*	Specifies the category to be summarized. *objCat* can be the object name, object identifier, or both.

Example

This example summarizes the Belgium category in the Europe dimension view.

```
SummarizeCat "Europe" Dimension "Locations" Category "Belgium"
```

SummarizeLevel

The `SummarizeLevel` verb summarizes the data values for the specified level in a view.

The Windows interface equivalent is to click the **Summarize** option on the **Diagram** menu when a level is selected.

The syntax is as follows:

```
SummarizeLevel objView [
Dimension objDim]
[Drillobjdrill]
Levels objLevel
```

Argument	Description
SummarizeLevel *objView*	Specifies the view. *objView* can be the object name, object identifier, or both.
Dimension *objDim*	Must be specified if the view is not referenced by object identifier. *objDim* can be the object name, object identifier, or both.
Drill *objDrill*	Is specified, if necessary, to uniquely identify the view. *objDrill* can be the object name, object identifier, or both.
Levels *objLevel*	Specifies the level to be summarized. *objLevel* can be the object name, object identifier, or both.

Example

This example summarizes the Branch Code level in the Europe dimension view.

```
SummarizeLevel "Europe" Dimension "Locations" Drill "By Region" Levels
"Branch Code"
```

UpdateForwardReference

The `UpdateForwardReference` verb may be used to resolve forward-referencing problems that can occur when you change a model by using a new verb in the model definition file. There is no Windows interface equivalent.

For more information about setting the `VerbOutput` option, see "Recommendation - Set Transformer to Verb MDL" (p. 18).

Processing times may be slower when you use the `UpdateForwardReference` verb.

The syntax is as follows:

```
UpdateFowardReference
```

UpdatePowerCubes

The `UpdatePowerCubes` verb allows you to update the metadata in the .mdc files for existing PowerCubes without updating the data.

The Windows interface equivalents are the **Update PowerCubes** and **Update Selected PowerCubes** options on the **Run** menu.

The syntax is as follows:

```
UpdatePowerCubes [objCube]
[updatepowercubeopts]
```

Argument	Description
UpdatePowerCubes *objCube*	Updates the PowerCube *objCube*. *objCube* can be the object name, object identifier, or both. If no PowerCube is specified, all are updated.

Argument	Description

updatepowercubeopts	Optional parameters that describe the PowerCube in greater detail. These options are:
	`DrillThrough {True|False}`
	`Objects {True|False}`
	`UserClasses {True|False}`
	`CurrencyConversion {True|False}`
	`DrillThrough` updates the drill through targets into the Cube. `Objects` updates the object names. `UserClasses` updates the security. `CurrencyConversion` updates the Currency Conversion tables.

Example

This example updates the metadata in the cube1.mdc file.

```
UpdatePowerCubes mycube.mdc UserClasses True
```

ViewAdd

The `ViewAdd` verb adds a dimension view. If the object already exists in the model, an error message is issued.

The Windows interface equivalent is to click **Add New View** in the menu option that is accessed by right-clicking the relevant portion of the view pane (**Diagram**).

The syntax is as follows:

```
ViewAdd objView [
Dimension objDim]
[Type viewtype]
[viewopts]
```

Argument	Description
ViewAdd *objView*	Adds the dimension view *objView*. *objView* must be the object name and can include the object identifier.
Dimension *objDim*	Is specified to locate the dimension view within a dimension. *objDim* can be the object name, object identifier, or both.
Type *viewtype*	Specifies the type of view. *viewtype* can be `All`, `Omit` or `Custom`. If not specified, the default is `Custom`.

Argument	Description
viewopts	Optional parameters that describe the view in greater detail. Some options are set by default if you do not set them. For the complete list of options, see viewopts.

Example

This example creates a Europe view for the Locations dimension and applies the Apex action. The `ViewSecurity` argument is blank (null), indicating that this is a dimension view.

```
ViewAdd "Europe" Dimension "Locations" ViewSecurity "" Apex "Europe"
```

For examples of custom views that use IBM Cognos 8 security objects rather than user classes, see the Security chapter of the Transformer *User Guide*.

ViewDelete

The `ViewDelete` verb removes a dimension view from the model.

The Windows interface equivalent is to click **Delete** in the menu that is accessed by right-clicking a dimension view.

You cannot delete a default view.

The syntax is as follows:

```
ViewDelete objView [
Dimension objDim]
```

Argument	Description
ViewDelete *objView*	Deletes the dimension view *objView*. *objView* can be the object name, object identifier, or both.
Dimension *objDim*	Must be specified if the view is not referenced by object identifier. *objDim* can be the object name, object identifier, or both.

Example

This example deletes the dimension view Europe.

```
ViewDelete "Europe" Dimension "Locations"
```

ViewListUpdate

The `ViewListUpdate` verb reorders a dimension view list.

The Windows interface equivalent is to drag the dimension views into the required order in the list.

You cannot move the default views. All Categories and Omit Dimensions always appear as the first and second items in the list.

The syntax is as follows:

ViewListUpdate [**Dimension** *objDim*] **StartList** *objViews***EndList**

Argument	Description
Dimension *objDim*	Must be specified if the views are not referenced by object identifier. *objDim* can be the object name, object identifier, or both.
StartList *objViews*	Lists the dimension views in the desired order. The dimension views can be identified by object name or object identifier.

Example

This example moves the views Far East and Europe to the top of the custom views in the dimension view list. The default views still always appear first.

```
ViewListUpdate Dimension "Locations" StartList "Far East" "Europe" EndList
```

ViewMake

The `ViewMake` verb creates a dimension view or updates an existing one.

The Windows interface equivalent, when an existing view is selected, is to click **Options** on the **Diagram** menu. For a new view, click the **Add New View** option in the menu that is accessed by right-clicking a dimension view.

The syntax is as follows:

ViewMake *objView* [
Dimension *objDim*]
[**Type** *viewtype*]
[*viewopts*]

Argument	Description
ViewMake *objView*	Creates the dimension view *objView* or modifies it if it exists. *objView* can be the object name, object identifier, or both. Include the object name if the dimension view does not exist.
Dimension *objDim*	When creating a dimension view, this specifies the dimension in which it is placed. When updating a view, this must be specified if the view is not referenced by object identifier. *objDim* can be the object name, object identifier, or both.

Argument	Description
Type {All|Omit|Custom}	Specifies the type of view. Type can be `All`, `Omit`, or `Custom`. If the view does not exist, the default is `Custom`.
viewopts	Optional parameters that describe the view in greater detail. Some options are set by default if you do not set them. For the complete list of options, see viewopts. If the view exists, previously set options are retained unless you change them with this command.

Example

This example defines a Europe view, which shows only the data for that region. The `ViewSecurity` argument is blank (null), indicating that this is a dimension view.

```
ViewMake "Europe" Dimension "Locations" ViewSecurity 0 Apex "Europe"
```

This example shows the two default views for the Products dimension. Transformer creates these default views for each dimension in the model. Again, `ViewSecurity` is blank or null.

```
ViewMake "All Categories" Dimension "Products" Type All ViewSecurity
0 ViewMake "Omit Dimension" Dimension "Products" Type Omit ViewSecurity 0
```

ViewUpdate

The **ViewUpdate** verb updates an existing dimension view.

The Windows interface equivalent is to click **Options** on the **Diagram** menu when an existing view is selected.

You cannot update the default views that Transformer creates for each dimension (Omit Dimension and All Categories).

For more information about updating views, see "ViewMake" (p. 194).

The syntax is as follows:

```
ViewUpdate objView [
Dimension objDim]
[Type viewtype]
[viewopts]
```

Argument	Description
ViewUpdate *objView*	Specifies the view to update. *objView* can be the object name, object identifier, or both.
Dimension *objDim*	Must be specified if the view is not referenced by object identifier. *objDim* can be the object name, object identifier, or both.

Argument	Description
Type*viewtype*	Specifies the type of view. *viewtype* can be `All`, `Omit`, or `Custom`. If not specified, the existing view type is retained.
viewopts	Optional parameters that describe the view in greater detail. For the complete list of options, see viewopts. Previously set options are retained unless you change them with this command.

Example

This example applies a summarize operation to the Belgium category in the Europe dimension view.

```
ViewUpdate "Europe" Dimension "Locations" Summary "Belgium"
```

Chapter 6: MDL Options

This section of the *Developer Guide* describes the Model Definition Language (MDL) options you can use with IBM Cognos Transformer to define and manipulate objects.

Each grouping applies to one or more Transformer verbs and the objects on which they operate. The option descriptions include some or all of the following:

- purpose of the option group
- the location of the equivalent functionality on the Windows interface (UI)
- verb syntax
- notes
- examples

Before you begin, please review the "Syntax Conventions" (p. 83) and the topic that explains when to use object identifiers or category codes: "Locating Objects Uniquely" (p. 24).

For more information about Transformer version 8.x features and functionality, see the Transformer *User Guide*.

appqueryopts

Use `appqueryopts` to set the options that apply to the following MDL verbs: `DataSourceAdd`, `DataSourceMake`, `DataSourceUpdate`, and `NewModel`.

In IBM Cognos Transformer version 8.x, you can use MDL to specify a package, report, or query from an IBM Cognos 8 data source and, optionally, any filter references.

AppInfo

The `AppInfo` option allows you to add a description that provides details about the query.

The syntax is as follows:

```
AppInfo string  [
string...]
```

where each string may be up to 256 characters and the total description may be up to 4,095 characters.

Example

```
AppInfo "This model created by DBX for HHY."
```

CharacterSet

The `CharacterSet` option specifies the type of character set used in the query.

The syntax is as follows:

CharacterSet *charset*

where *charset* is one of `Default`, `ANSI`, `OEM`, `Unicode` or `Multibyte`.

Example

```
CharacterSet Multibyte
```

CognosSource

The `CognosSource` option specifies an IBM Cognos 8 package or report data source in the model. All package or report definitions are added before query definitions.

The syntax is as follows:

CognosSource id *"name"*

where *name* is the name of the package in the Transformer model. All packages, reports, and queries must have unique names.

Example

```
CognosSource 103 "Go Sales and Retailers"
```

Columns

The `Columns` option adds information about the type of column used by the data source.

This option relates to the structure of the source data and should not be manually set.

The syntax is as follows:

Columns {True|False}

where `False` indicates that the data is fixed length.

Example

```
Columns True
```

DataRange

The `DataRange` option adds information about the table ranges used from the source data.

This option applies only to queries made against Excel databases.

The syntax is as follows:

DataRange *string*

where *string* is the name of the Excel data range.

Example

```
DataRange "reptotal_rng"
```

DecimalSep

The `DecimalSep` option specifies the character used as the decimal separator for the data values.

The syntax is as follows:

DecimalSep *string*

where *string* is the decimal separator.

Example

```
DecimalSep "."
```

ImrName

The `ImrName` option specifies the drill-through target for an IQD data source.

This option is required for drill-through to an Impromptu report from an IQD data source.

The syntax is as follows:

`ImrName` *`string`*

where *string* is a file name and optional path.

Example

```
ImrName "C:installation_directory\PowerCubes and Reports\bsc_msrs.imr"
```

Isolation

The `Isolation` option provides information about the IQD data source.

This option applies only to IQD data sources.

The syntax is as follows:

`Isolation` *`number`*

where *number* is a number from `0` to `6` that corresponds to the following list:

Isolation Level	*number*
Default	0
Read Uncommitted	1
Read Committed	2
Cursor Stability	3
Reproducible Read	4
Phantom Protection	5
Serializable	6

Example

```
Isolation 2
```

ModelStamp

The `ModelStamp` option assigns a numeric code to the model.

This option is for internal use only.

The syntax is as follows:

ModelStamp *bignum*

Example

```
ModelStamp 827358629
```

PackageReportSource

The `PackageReportSource` option specifies a reference to the package on which an IBM Cognos 8 query is defined.

PackageReportSource *source ID "source name"*

Example

```
PackageReportSource 103 "Go Sales and Retailers"
```

PackageTimeStamp

The `PackageTimeStamp` option specifies the version of the package last used.

The syntax is as follows:

PackageTimeStamp *"pathand time stamp"*

Examples

```
PackageTimeStamp "/content/package[@name='GO Sales and Retailers200701011200']/model[@name='model']"
```

```
PackageTimeStamp "/content/package[@name='Go Sales']/model[@name='2008-04-16T15:41:14.078Z']
```

PreSummarized

The `PreSummarized` option specifies whether measures are externally rolled up before they are brought into the model.

The syntax is as follows:

PreSummarized {True|False}

Example

```
PreSummarized False
```

SegmenterPrompt

The `SegmenterPrompt` option specifies the prompt used to segment a SAP BW fact query during stream extract.

The syntax is as follows:

SegmenterPrompt*string*

where *string* is the name of the prompt used for segmenting the query.

Example

```
SegmenterPrompt "WhichBrand"
```

SegmenterPromptEnabled

The `SegmenterPromptEnabled` option specifies whether stream extract is to be used for eligible SAP BW fact queries.

The syntax is as follows:

`SegmenterPromptEnabled` `{True|False}`

Example

```
SegmenterPromptEnabled True
```

Separator

The `Separator` option specifies the character used to separate character-delimited fields in the data source.

This option applies only to sources that use delimited-field text or delimited-field text with column titles.

The syntax is as follows:

`Separator` *`string`*

Example

```
Separator ","
```

SetCurrent

The `SetCurrent` option specifies whether the source data sets the current date in the time dimension for the model.

The syntax is as follows:

`SetCurrent` `{True|False}`

Example

```
SetCurrent False
```

Source

The `Source` option specifies the name and location of the source file for the model.

The syntax is as follows:

`Source` *`string`*

where *string* is a file name and optional path.

Example

```
Source "c:installation_directory\cubes and reports\prodinfo.csv"
```

SourceInfo

The `SourceInfo` option specifies additional descriptive information about the source file for the model.

The syntax is as follows:

SourceInfo *string* [
string...]

where each string may be up to 256 characters, and the total description may be up to 4,095 characters.

Example

```
SourceInfo "This data source created by DBX."
```

SourcePath

The `SourcePath` option specifies the path to the IBM Cognos 8 package or report within Content Manager.

The syntax is as follows:

SourcePath *"path"*

Example

```
SourcePath "/content/package [@name='Go Sales and Retailers']"
```

SourceSignonList

The `SourceSignonList` option specifies the signons that apply to the data source.

The syntax is as follows:

SourceSignonList *signonlist* **EndList**

where *signonlist* is the list of signons for a data source, where these are required. Signons can be the object name, object identifier, or both.

Example

```
SourceSignonList 105 EndList
```

SourceType

The `SourceType` option allows you to specify any supported data source for the model.

For an IQD data source, the keyword is `DataSource`. However, for an IBM Cognos 8 single query, this keyword must be qualified by specifying `SourceType CognosSourceQuery`.

Similarly, the keyword for an IBM Cognos 8 package or report is `CognosSource`. It must be qualified by specifying `SourceType package` or `SourceType report`.

The syntax is as follows:

SourceType *sourcetype*

where *sourcetype* is one of `FlatFileColNames`, `FlatFile`, `DataSource`, `Access`, `AccessQuery`, `ExcelCrosstab`, `ExcelDatabase`, `PowerHousePortable`, `FixedASCII`, `FixedASCIINoCrlf`,

Dictionary or, for IBM Cognos 8 data sources, one of either CognosSourceQuery, package, or report.

Example

The following example specifies an IBM Cognos 8 package as the source type:

```
CognosSource 103 "GO Sales" SourceType package SourcePath "/content/package
[@name='GO Sales']"
```

The following example specifies an IBM Cognos 8 report as the source type:

```
CognosSource 104 "GO Sales" SourceType report SourcePath "/content/package
[@name='GO Sales']/report@name='GO SalesReport']"
```

The following example specifies a single query created using an IBM Cognos 8 report as the source:

```
DataSource 116 "Time" Separator ","SourceType
CognosSourceQuery CharacterSet Default DecimalSep ".
" ThousandSep "," ColumnsTrue Timing PopYesCreateDefault

PackageReportSource104 "GO Sales
Report" SetCurrent True Speed FalsePresummarized False
```

Speed

The Speed option allows you to specify whether query optimization or uniqueness verification checks are enabled for your model.

The syntax is as follows:

```
Speed {True|False}
```

where True enables the Maximize Data Access Speed processing option. False enables the Verify Category Uniqueness processing option.

Example

```
Speed False
```

SQL

The SQL option represents the query as a Structured Query Language (SQL) string.

This option cannot be changed by the user. Any changes must be made in the source.

The apostrophe (') is the delimitation character for any SQL statement. It must appear at the beginning and end of every line of SQL.

The maximum number of characters on any line is 256.

The syntax is as follows:

```
SQL' string [
string...]'
```

Example

The following represents the SQL for the Main IQD data source:

```
SQL 'select T1."ORDER_DT" as c1,' 'T2."PROD_NO" as
c2, T1."REP_NO" as c3,'
'T1."CUST_NO" as c4, T2."QTY" as c5,'
'(T2."QTY" * T2."PRI' 'CE") as c6,'
```

```
'(T2."QTY" * T3."PROD_COST") as c7,'
'(CASE '
'WHEN ((((T2."QTY" * T2."PRICE")-'
'(T2."QTY" * T3."PROD_COST"))/(T2."QTY" * '
'T2."PRICE")) <= 0.19) '
'THEN ("'Under 20%')'
'WHEN ((((T2."QTY" * T2."PRICE") - '
'(T2."QTY" * T3."PROD_COST")) / '
'(T2."QTY" * T2."PRICE")) BETWEEN 0.2 AND ' '0.65)'
'THEN ("'20% - 65%')'
'WHEN ((((T2."QTY" * T2."PRICE") - '
'(T2."QTY" * T3."PROD_COST"))'
'/ (T2."QTY" * T2."PRICE")) >= 0.66)'
'THEN (' "'Over 65%')'
'ELSE ('ERROR')'
'END) as c8 from "ORDER" T1,'
'("PRODUCT" T3 left outer join "ORDRDETL" T2 '
'on T2."PROD_NO" = T3."PROD_NO") where'
'(T2."ORDER_NO" = T1."ORDER_NO")'
'order by c1 asc,c3 asc,c4 asc,c2 asc'
```

Stamp

The `Stamp` option assigns a numeric code to the model for internal use.

This option is for internal use only.

The syntax is as follows:

Stamp *number*

Example

```
Stamp 890332424
```

StreamExtractSize

The `StreamExtractSize` option specifies the size of the stream extract buffer used for an SAP BW fact query.

The syntax is as follows:

StreamExtractSize*number*

where number is the size in MB of the stream extract buffer, or 0. The number 0 disables stream extracts for the query.

Example

```
StreamExtractSize 10
```

SuppressNull

The `SuppressNull` option specifies whether to suppress null values in IBM Cognos 8 packages based on a SAP BW data source.

The syntax is as follows:

SuppressNull*suppressnull*

where *suppressnull* is one of SuppressNullYes, SuppressNullNo, or SuppressNullModel.

Example

```
SuppressNull SuppressNullYes
```

ThousandSep

The `ThousandSep` option specifies which character demarcates numbers in the thousands, or larger.

The syntax is as follows:

`ThousandSep` *`string`*

Example

```
ThousandSep ","
```

Timing

The `Timing` option specifies when categories are generated during cube-building.

The syntax is as follows:

`Timing`*`datasourcetiming`*

where *datasourcetiming* is one of `PopNoCreateNo`, `PopYesCreateNo`, `PopNoCreateDefault`, `PopYesCreateDefault`, `PopNoCreateYes`, `PopYesCreateYes`. These, in turn, relate to the following Windows interface settings:

datasourcetiming	Windows interface setting
`PopNo`	**Generate Categories** is not selected
`PopYes`	**Generate Categories** is selected
`Create Default`	**PowerCube Creation** and **Default** are selected
`CreateYes`	**PowerCube Creation** and **Create the PowerCubes** are selected
`CreateNo`	**PowerCube Creation** and **Generate Categories Only** are selected

Example

```
Timing PopYesCreateNo
```

UpdateCycle

The `UpdateCycle` option is set by Transformer for internal use only.

The syntax is as follows:

`UpdateCycle` *`bignum`*

Version

The `Version` option is set by Transformer. It should not be set manually.

The syntax is as follows:

Version *string*

Example

```
Version "7.4 1012"
```

assocopts

Use `assocopts` to set the options that apply to the following MDL verbs: `AssociationAdd`, `AssociationMake`, `AssociationUpdate`, `CurrencyTableAdd`, `CurrencyTable Make`, `CurrencyTableUpdate`, `DimensionAdd`, `DimensionMake`, `DimensionUpdate`, `LevelAdd`, `LevelMake`, `LevelUpdate`, `MeasureAdd`, `MeasureMake`, and `MeasureUpdate`.

AssociationContext

The `AssociationContext` option specifies the context for an object in the overall model.

The syntax is as follows:

AssociationContext *objcontext*

where *objcontext* can be the object name or object identifier or both.

Example

```
AssociationContext 2979 "Branch Code"
```

AssociationReferenced

The `AssociationReferenced` option specifies the reference used for the association.

The syntax is as follows:

AssociationReferenced *objref*

where *objref* is the name of the association

Example

```
AssociationReferenced "Branch Code"
```

AssociationRole

The `AssociationRole` option specifies the role played by the associated object in the model.

The syntax is as follows:

AssociationRole *objrole*

where *objrole* is one of `Role_Catcode`, `Role_Description`, `Role_Drillthrough`, `Role_Label`, `Role_Orderby`, `Role_Source`, or `Role_Tag`.

For Currency tables, *objrole* is one of `Role_CountryCode`, `Role_Date`, `Role_Label`, or `Role_Rate`.

Example

```
AssociationRole Role_Label
```

AssociationType

The `AssociationType` option specifies the type of object represented by the association.

The `Type_Query` option is used for Impromptu Query Definition (IQD) data sources.

The syntax is as follows:

`AssociationType` *`objtype`*

where *objtype* is one of `Type_PowerCube` or `Type_Query`.

Example

```
AssociationType Type_PowerCube
```

catopts

Use `catopts` to set the options that apply to the following MDL verbs: `CatAdd`, `CatMake`, `CatUpdate`, `SpecialCatAdd`, `SpecialCatMake`, `SpecialCatUpdate`, `DrillCatMake`, `RootCatMake`, `RootCatUpdate`, `SubDimRootMake`, and `SubDimRootUpdate`.

Blanks

The `Blanks` option indicates the presence of missing values for regular categories.

For more information about how null and missing values are treated in the supported IBM Cognos 8 reporting applications, see the Measures chapter of the Transformer *User Guide*.

This option applies to regular categories only.

The syntax is as follows:

`Blanks` `{True|False}`

where true means the category is blank (has missing values).

Example

```
Blanks False
```

Calc

The `Calc` option specifies a calculation and the model objects to which it applies.

The Calculation keywords supported by Transformer are as follows:

Calculation keyword	Type	Description
/	`Number`	Divides.
-	`Date` or `Number`	Subtracts.
+	`Character`, `Date`, or `Number`	Adds.

Calculation keyword	Type	Description
^	Number	Returns a number raised to the power of a second number.
Accumulate	Number	Computes the running total.
AccPOB	Number	Computes the running total as a percentage of the total.
Average	Number	Averages.
Max	Number	Represents a maximum instance.
Min	Number	Represents a minimum instance.
*	Number	Multiplies.
Percent-Growth	Number	Computes the percentage change that parameter 2 is, relative to parameter 1.
Share	Number	Computes the share that parameter 2 is, relative to parameter 1.
Change	Number	Computes the difference between two parameters (parameter 2 minus parameter 1)

The syntax is as follows:

Calc *calculation*

where *calculation* is a combination of *objects* and *keywords*.

objects consist of the object name followed by the at sign (@), the object type and, optionally, the object identifier. The object type can be Category, Level, or Drill, such as GO Water Bottle@Category@4805.

Example

The following examples represent two commonly encountered calculation specifications:

```
Calc Percent-Growth ("Prior YTD","YTD")
Calc Share ("GO Water Bottle@Category@4805, "Sport Wear@Category@4799")
```

CalcDef

The CalcDef option references a calculation defined elsewhere in the model.

The `CalcDef` option can serve as a forward reference for the `DimCalcDef` object. A `CatMake` statement that references a calculation can appear in the MDL file before the calculation object is defined using the `DimCalcDefMake` statement.

The syntax is as follows:

`CalcDef` *`objDimCalcDef`*

where *objDimCalcDef* can be the object name or object identifier.

Example

```
CalcDef 6375
```

CatInfo

The `CatInfo` option adds a category description to the model.

The `CatInfo` option describes regular, special, drill, and root categories.

The syntax is as follows:

`CatInfo` *`string`* [
`string...`]

where each string may be up to 256 characters and the total description may be up to 4,095 characters.

Example

```
CatInfo "Exceptional money-makers! Determine why these items are profitable
and use the information to improve the performance of others in the group."
```

ContextLevel

The `ContextLevel` option specifies the level on which a relative time period is based, establishing the context for the target period.

The *string* parameter is context-sensitive.

The syntax is as follows:

`ContextLevel` *`string`*

where *objDimCalcDef* can be the object name or object identifier.

Example

```
ContextLevel "Quarter"
```

ContextOffset

The `ContextOffset` option specifies the number of periods by which the relative time period is offset (or removed) from the current period, where 0 corresponds to the current period. For grouped categories, this value defined the range of relative time categories in the group.

The syntax is as follows:

`ContextOffset` *`number`*

Example

```
ContextOffset -1
```

Current

The `Current` option specifies the date category that will serve as the current period.

The syntax is as follows:

`Current`

Example

```
Current
```

Date

The `Date` option specifies the start date for a date category. This option is required.

The syntax is as follows:

`Date` *`date`*

where *date* is the start of the date period.

Example

```
Date 19990101
```

DateDrill

The `DateDrill` option specifies the drill category for the time dimension.

The syntax is as follows:

`DateDrill` *`objDrillCat`*

where *objDrillCat* is the object name or object identifier.

Example

```
DateDrill "By Time"
```

ExtraWeek

The `ExtraWeek` option specifies the conditions under which Transformer adds extra days as a separate week, to resynchronize a lunar quarter or year with the regular calendar. You can specify that an extra week never be added, or only added if there are exactly 7 extra days. Or, you can specify some other condition that will trigger the addition of an extra week, such as 4 or more remainder days.

The syntax is as follows:

`ExtraWeek` *`extraweek`*

where *extraweek* is one of `None`, `7only`, `, 6orMore5orMore`, or `4orMore`.

Example

```
ExtraWeek 4orMore
```

Filtered

The `Filtered` option specifies whether categories are excluded from the model.

The syntax is as follows:

`Filtered` `{True|False}`

Example

```
Filtered True
```

Format

The `Format` option specifies the number of decimal places in a numeric measure.

The syntax is as follows:

`Format` *`string`*

where *string* is the format, followed by a tilde (~) and the number of decimal places.

Example

This example sets the format to #,##0 and sets the decimal places to 2:

```
Format "#,##0~2"
```

Inclusion

The `Inclusion` option specifies whether categories are included or suppressed.

This parameter applies to regular, special, and drill categories.

The syntax is as follows:

`Inclusion`*`inclusion`*

where *inclusion* is one of `Default`, `Generate`, `Suppress`, `Retain`, or `Filtered`.

The Windows interface equivalents of these are as follows:

- **Default** (when needed)
- **Always Include**
- **Suppress** (on the **Diagram** menu)
- **Include When Needed** or **Suppress Blank Categories**
- **Exclude** (on the **Diagram** menu)

Example

This example suppresses the specified category:

```
Inclusion Suppress
```

IsKeyOrphanage

The `IsKeyOrphanage` option specifies whether the category in a manual level is a placeholder that will serve as the parent category for any newly generated categories that appear in the next lower source level.

Only one orphanage is allowed in a lower manual level for each source level category.

For easier reference, you can specify that the category code of the orphanage be shown for a particular category.

The syntax is as follows:

`IsKeyOrphanage` `{True|False}`

Example

```
IsKeyOrphanage False
```

IsTruncated

The `IsTruncated` option specifies whether an overly long category identifier has been truncated.

The syntax is as follows:

`IsTruncated` `{True|False}`

Example

```
IsTruncated True
```

Label

The `Label` option adds a descriptive name to clearly identify a category.

The syntax is as follows:

`Label` *`stringwithnewline`*

Example

```
Label "Day Tripper"
```

LastUse

The `LastUse` option specifies the date when a category was created or last updated in the model.

The syntax is as follows:

`LastUse` *`date`*

where *date* is in the format `YYYYMMDD`.

Example

```
Lastuse 20070324
```

NewPartition

The `NewPartition` option specifies the partition level for a category.

The syntax is as follows:

NewPartition *uns*

where *uns* is an unsigned number that represents the partition level of that category.

Example

```
NewPartition 1
```

HideValue

The `HideValue` option hides meaningless measure values, either at the root or at any other category level in a scenario dimension.

We recommend that you use this option with `DimDefaultCategory` to specify a new default category at a lower level, having meaningful measure values, where you want the cube to open.

The syntax is as follows:

HideValue `{True|False}`

Example

```
RootCatUpdate "Products" Dimension "Products" Inclusion Generate ...
 HideValue True
```

Orphanage

The `Orphanage` option is automatically set by Transformer when a manual category is added. It is for internal use only.

The syntax is as follows:

Orphanage

Example

```
Orphanage
```

PartialWeek

The `PartialWeek` option specifies how partial-week categories are to be allocated or split when they span two higher-level time periods.

The syntax is as follows:

PartialWeek *partialweek*

where *partialweek* is one of `First`, `Last`, `Largest`, `Split`, `SplitIfGreater`, or `None`. The `None` parameter does not have a Windows interface equivalent. The Windows interface equivalents of the other parameters are as follows:

- **First Period**
- **Last Period**
- **Largest Period**
- **Always Split**
- **Split > 1 Day**)

Example

This example splits the categories into two weeks if there is more than one day in the extra partial week:

```
PartialWeek SplitIfGreater
```

Primary

The `Primary` option specifies the primary drill category for a set of categories.

The syntax is as follows:

`Primary` *`objCat`*

where *objCat* can be the object name or object identifier.

Example

```
Primary "20071201-20071231"
```

PrimaryDrill

The `PrimaryDrill` option specifies whether the category is in the primary drill path.

The syntax is as follows:

`PrimaryDrill` `{True|False}`

Example

```
PrimaryDrill True
```

Rollup

The `Rollup` option specifies whether the measure values from the category should be rolled up, or summarized.

The syntax is as follows:

`Rollup` `{True|False}`

Example

```
Rollup True
```

RunningPeriods

The `RunningPeriods` options specifies the number of relative time periods for an N-period running total.

The syntax is as follows:

`RunningPeriods` *`uns`*

where *uns* is the number of running periods to include.

Example

```
RunningPeriods 0
```

Share

The `Share` option applies a Transformer-specific calculation that computes the proportionate share of an item relative to its parent category.

For more information, see the Transformer *User Guide*.

The syntax is as follows:

Share *objCat*

where *objCat* is the object name or object identifier.

Example

```
Share "Back Packs"
```

ShortName

The `ShortName` option specifies a shorter name for the category.

You can use this option with regular, special, drill, and root categories.

The syntax is as follows:

ShortName *string*

Example

```
ShortName "ENV"
```

Sign

The `Sign` option specifies whether the sign of a measure value associated with a category is to be reversed, changing it from a negative to a positive number, or from a positive to a negative number.

This option is useful if you need to convert values into the form required by specific financial reports, such as debits and credits, or assets and liabilities.

You can use this option with regular and special categories.

The syntax is as follows:

Sign {True|False}

where True means that the sign is reversed.

Example

```
Sign True
```

SourceValue

The `SourceValue` option specifies the name by which a source column is identified.

This option applies to regular categories only.

The syntax is as follows:

SourceValue *stringwithnewline*

Example

```
SourceValue "Environmental Line"
```

SplitWeek

The `SplitWeek` option specifies whether extra days should be divided between two weeks.

This option is for internal use only.

The syntax is as follows:

`SplitWeek` `{True|False}`

Example

```
SplitWeek False
```

Suppressed

The **Suppressed** option specifies whether the category should have its values suppressed (hidden).

You can use this option with regular and special categories.

The syntax is as follows:

`Suppressed` `{True|False}`

where True means that the category values are suppressed (hidden).

Example

```
Suppressed True
```

TargetLevel

The `TargetLevel` option specifies the target category for a relative time definition or, for to-date or N-period running totals, the granularity of periods that you set to compute the totals.

The syntax is as follows:

`TargetLevel` *`string`*

Example

```
TargetLevel "Month"
```

TargetOffset

The `TargetOffset` option sets the number of periods by which the relative time category is offset (or removed) from the current period. The effect of your selection depends on whether you are creating a relative time category for a Single Category, a Period To-Date, or an N-Period Running Total.

The syntax is as follows:

`TargetOffset` *`number`*

Example

```
TargetOffset -1
```

TimeAggregate

The `TimeAggregate` option specifies the type of relative time, such as To-Date, Running Total, or a grouped version of these.

The syntax is as follows:

`TimeAggregate`
`timeaggregate`

where *timeaggregate* is one of `None`, `Single`, `ToDate`, `ToDate_Grp`, `Running`, or `Running_Grp`.

Example

```
TimeAggregate ToDate
```

ToDateLevel

The `ToDateLevel` option specifies the level to which the Period To-Date relative time category applies.

The syntax is as follows:

`ToDateLevel` *`objLevel`*

where *objLevel* is the object name.

Example

```
ToDateLevel "Quarter"
```

WeekBegins

The `WeekBegins` option specifies the day that each week in the model begins.

The syntax is as follows:

`WeekBegins` *`day`*

where *day* is one of `Sunday`, `Monday`, `Tuesday`, `Wednesday`, `Thursday`, `Friday`, or `Saturday`.

Example

```
WeekBegins Sunday
```

YearBegins

The `YearBegins` option specifies the date that the fiscal year begins, where this differs from the start of the calendar year.

The syntax is as follows:

`YearBegins` *`date`*

where *date* is in the format `YYYYMMDD`

Example

```
YearBegins 20070401
```

cognospackageopts

Use `cognospackageopts` to set the options that apply to the following MDL verbs: `CognosPackageAdd`, `CognosPackageMake`, and `CognosPackageUpdate`.

Description

The `Description` option adds a description of the source column associated with the package.

The syntax is as follows:

`Description` *`objCol`*

where *objCol* is the object name of a column.

Example

```
Description "Product types"
```

PackageTimeStamp

The `PackageTimeStamp` option specifies the version of the package last used.

The syntax is as follows:

`PackageTimeStamp`*`"pathand time stamp"`*

Examples

```
PackageTimeStamp "/content/package[@name='GO Sales and Retailers200701011200']/model[@name='model']"
```

```
PackageTimeStamp "/content/package[@name='Go Sales']/model[@name='2008-04-16T15:41:14.078Z']
```

SourcePath

The `SourcePath` option specifies the path to the IBM Cognos 8 package or report within Content Manager.

The syntax is as follows:

`SourcePath` *`"path"`*

Example

```
SourcePath "/content/package [@name='Go Sales and Retailers']"
```

SourceType

The `SourceType` option specifies which type of metadata source is being referenced in Content Manager. The valid values are package or report.

The syntax is as follows:

`SourceType`*`sourcetype`*

where *sourcetype* is one of `package` or `report`.

Example

The following example specifies an IBM Cognos 8 package as the source type:

```
CognosSource 103 "GO Sales" SourceType package SourcePath "/content/package
[@name='GO Sales']"
```

The following example specifies an IBM Cognos 8 report as the source type:

```
CognosSource 104 "GO Sales" SourceType report SourcePath "/content/package
[@name='GO Sales']/report@name='GO SalesReport']"
```

colopts

Use `colopts` to set the options that apply to the following MDL verbs: `ColumnAdd`, `ColumnMake`, `ColumnUpdate`, `DataSourceAdd`, `DataSourceMake`, `DataSourceUpdate`, `SourceAttributeAdd`, `SourceAttributeMake`, and `SourceAttributeUpdate`.

Calc

The `Calc` column option defines the calculation and the objects involved in the calculation.

The syntax is as follows:

`Calc`*`keywordsobjects`*

where *objects* are the object name followed by the at sign (`@`) and the object identifier; for example, `"Revenue@259"`

The keywords used with the `Calc` column option are as follows:

Calculation keyword	Type	Description
/	Number	Divided by
=	Boolean	Equals
>	Boolean	Greater than
>=	Boolean	Greater than or equal to
<	Boolean	Less than
<=	Boolean	Less than or equal to
(		Left parenthesis
-	Date or Number	Minus
*	Number	Multiplied by
+	Character, Date, or Number	Adds numbers or concatenates strings

Calculation keyword	Type	Description
^	Number	Returns a number raised to the power of a second number
)		Right parenthesis
Absolute	Number	Converts numbers to their unsigned value
Add-Days	Date	Returns the date resulting from adding a number of days to a date
Add-Months	Date	Returns the date resulting from adding a number of months to a date
Add-Years	Date	Returns the date resulting from adding a number of years to a date
Age	Date	Age
Ceiling	Number	Returns a number rounded to the next highest integer
Char-Length	Number	Returns the number of characters in a string
Date-To-Days-From-1900	Number	Returns the number of days from January 1, 1900 inclusive
Day	Number	Returns the day of the month (1-31) from a date
Days-From-1900-To-Date-Time	Date	Returns the date obtained from converting a number of days from January 1, 1900 inclusive to a date
Days-To-End-Of-Month	Date	Returns the number of days to the last day of the month represented by a date
Else	Boolean	Else
First-Of-Month	Date	Returns the first day of a month from a date or datetime
First-Word	Character	Returns the first word of a string

Calculation keyword	Type	Description
Floor	Number	Returns a number rounded down to the next lowest integer
If	Boolean	If
Integer-Divide	Number	Returns the integer obtained by truncating the result of an integer divided by a second integer
IsNull	Character, Date, or Number	Is null
Last-Of-Month	Date	Returns the last day of a month from a date or datetime
Left	Character	Returns a specific number of characters, starting at the left of the string
Lower	Character	Converts uppercase characters to lowercase
Mod	Number	Returns the remainder (modulus) of an integer divided by a second integer
Month	Number	Returns the month number from a date, datetime, or interval
Months-Between	Number	Returns the integer number of months from a date to a second date
Not	Boolean	Not
Number-To-String	Character	Returns the string representation of a number rounded to the next integer
Or	Boolean	Or
Position	Number	Returns the starting position of a string in a second string
Reverse	Character	Returns a string with the order of the characters reversed

Calculation keyword	Type	Description
Right	Character	Returns a specific number of characters, starting at the right of the string
Round-Down	Number	Returns a number rounded down
Round-Near	Number	Returns a number rounded to the nearest integer
Round-Up	Number	Returns a number rounded up
Round-Zero	Number	Returns a number rounded towards zero
SqRt	Number	Returns the square root of a number
String-To-Number	Number	Converts a string to a number
SubString	Character	Returns a substring from a string
Then	Boolean	Then
Today	Date	Returns the current date according to your computer clock
Trim-Leading	Character	Returns a string with leading spaces removed
Trim-Trailing	Character	Returns a string with trailing spaces removed
Upper	Character	Converts lowercase characters to upper-case
Year	Number	Returns the year from the date
Years-Between	Number	Returns the number of years from a date to another date

Example

```
Calc "Order Qty" * "Price"
```

Class

The `Class` option specifies the data class associated with the column.

The syntax is as follows:

Class [*dataclass*]

where *dataclass* is one of `Default` or no setting, `Description`, `Date`, `Quantity`, `Ignore`, or `Member`. The Windows interface equivalents are:

- **Unspecified**
- **Text**
- **Date**
- **Numeric**
- **Ignore**
- **Array Member**

Example

```
Class Quantity
```

ColSrcType

The `ColSrcType` option defines the type of IBM Cognos 8 data source for the column.

This option is for internal use only.

The syntax is as follows:

ColSrcType*colsrctype*

where *colsrctype* is one of SAPMeasure, SAPBusinessKey, SAPLevelKey, SAPOther, RLDimensional, OLAPDIMENSIONAL, or SAPCalcmeasure.

Example

```
ColSrcType RLDimensional
```

Column

The `Column` option specifies the original name for the column in the data source.

You must specify the column.

If you change a column name in MDL, the object name may differ from the one on the Windows interface. This is because MDL takes the object name from the **Original Name** box on the Column property sheet, whereas Transformer takes the object name from the **Column Name** box of the Column property sheet.

The syntax is as follows:

Column *objCol*

where *objCol* is the object name.

Example

```
Column "Product ID"
```

ColumnInfo

The `ColumnInfo` option provides a description of the column in the model.

The syntax is as follows:

ColumnInfo *string* [
string...]

where each string may be up to 256 characters and the total description may be up to 4,095 characters.

Example

```
ColumnInfo "Gross revenue from product sales"
```

Dateconstant

The `Dateconstant` option must be inserted before any date constant used in a calculation.

The syntax is as follows:

Dateconstant '
date'

where each string may be up to 256 characters and the total description may be up to 4,095 characters.

Example

```
months-between (dateconstant '2007-01-01', today)
```

DateLevel

The `DateLevel` option specifies the level of the time dimension associated with the column.

The syntax is as follows:

DateLevel *datelevel*

where *datelevel* is one of `None`, `Year`, `Quarter`, `Month`, `Week`, or `Day`.

Example

```
DateLevel Day
```

Decimals

The `Decimals` option specifies the number of decimal spaces in the source data.

You should set this option to the actual value in the data source.

You can also set this option on the Windows interface, by using the **Decimal Places** box on the **Format** tab of the Measure property sheet.

The syntax is as follows:

Decimals *uns*

Example

```
Decimals 2
```

Detail

The `Detail` option specifies the degree of detail for the column.

You can select from a context-sensitive list of valid codes in the Windows interface.

For example, if you do not want to include all of the categories in a level, you can set the lowest level of detail to include. That way, you can avoid unwanted entries or duplicate names that need to be qualified by the name of their owning category, such as Brand (OF Tents).

The syntax is as follows:

`Detail` *`string`*

where *string* can be `"Unspecified"`

Example

```
Detail "Unspecified"
```

Format

The `Format` option specifies the date input format used by the date values in the column that populates the time dimension.

The syntax is as follows:

`Format` [*`dateformat`*]

where *dateformat* is one of `PreDefined`, `YMD`, `DMY`, `MDY`, `YM`, `MY`, `Y`, `Q`, `M`, `W`, or `D`.

Example

```
Format YMD
```

InputScale

The `InputScale` option specifies the number used to scale the data values in the column.

The syntax is as follows:

`InputScale` *`number`*

where *number* is not greater than `16`.

Example

```
InputScale 1
```

Offset

The `Offset` option specifies the number by which the column position is offset.

The syntax is as follows:

`Offset` *`uns`*

Example

```
Offset 2
```

Origin

The `Origin` option specifies the type of data source used for the column.

You should not manually set this option.

The syntax is as follows:

`Origin` *`{Source|Generated|Calculated|Manual}`*

where `Generated` is the default; `Source` specifies an IQD data source; `Calculated` specifies a calculated column; and `Manual` specifies a manually created column in the model.

Example

```
Origin Generated
```

Scale

The `Scale` column option specifies the output scale.

This option is either read directly from the data source or specified in the measure definition. For more information, see "Scale" (p. 252).

The syntax is as follows:

`Scale` *`uns`*

where *uns* is an unsigned number that specifies the output scale.

Example

```
Scale 0
```

Size

The `Size` column option specifies the size of certain types of data source.

This column option only applies when the `SourceType` of the associated data source is `PowerHousePortable`, `FixedASCII`, or `FixedASCIINoCrlf`.

The syntax is as follows:

`Size` *`uns`*

Example

```
Size 1
```

Storage

The `Storage` option specifies the actual storage type used by the source data for the column.

This column option relates directly to the source data.

The syntax is as follows:

`Storage`*`storagetype`*

where *storagetype* is one of `Default`, `Int8`, `Int16`, `Int32`, `Float64`, `Text`, `Character`, or `LongText`.

Example

```
Storage Int32
```

TimeArray

The `TimeArray` option specifies the type of array used to group items for the time dimension.

The syntax is as follows:

`TimeArray` *`timearray`*

where *timearray* is one of `Off`, `Quarter`, `Month`, `Month13`, `Quarter6`, `Month18`, `Quarter8`, or `Month24`.

Example

```
TimeArray Month13
```

TimeArrayCol

The `TimeArrayCol` option specifies the name of the array used to group items for the time dimension.

The syntax is as follows:

`TimeArrayCol` *`string`*

Example

```
TimeArrayCol "Time"
```

TimeArrayMonth

The `TimeArrayMonth` option specifies the month number on which the time array is based.

The syntax is as follows:

`TimeArrayMonth` *`uns`*

Example

```
TimeArrayMonth 2
```

currencyrecordopts

Use `currencyrecordopts` to set the options that apply to the following MDL verbs: `CurrencyAdd`, `CurrencyMake`, and `CurrencyUpdate`.

BaseCurrency

The `BaseCurrency` option defines the default currency stored in the Currency Table for the model.

The syntax is as follows:

`BaseCurrency`

where the Windows interface equivalent is to specify a <Base Default> name in the **Currency Record** dialog box, accessible by clicking the **Properties** button on the **Currency Table** dialog box.

Example

```
BaseCurrency
```

CountryCode

The `CountryCode` option specifies the 3-character international code used to represent each country listed in the Currency Table.

The syntax is as follows:

`CountryCode` *`code`*

where *code* is a unique identifier for a country.

Example

```
CountryCode "USA"
```

CurrencyCountryLabel

The `CurrencyCountryLabel` option specifies the name used to uniquely identify each country listed in the Currency Table.

The syntax is as follows:

`CurrencyCountryLabel` *`label`*

where *label* is a unique identifier for a country.

Example

```
CurrencyCountryLabel "Canada"
```

CurrencyDecimals

The `CurrencyDecimals` option specifies the number of decimal places used when displaying the converted currency values for each country listed in the Currency Table. However, this value may not be the same as the decimal place setting used in your reporting application. Depending on your implementation, it may be overridden from the Measure property sheet, or at runtime.

The syntax is as follows:

`CurrencyDecimals` *`decimals`*

where *decimals* can be a number from `0` to `9`. The default used with the Great Outdoors.mdl sample is `0`. The Transformer Currency Table default is `2`.

Example

```
CurrencyDecimals 5
```

CurrencyFormatOverride

The `CurrencyFormatOverride` option specifies whether the currency format settings can be overridden elsewhere in the IBM Cognos 8 modeling or reporting applications.

The syntax is as follows:

`CurrencyFormatOverride` `{True|False}`

Example

```
CurrencyFormatOverride True
```

CurrencyIsEMU

The `CurrencyIsEMU` option specifies whether the country is a member of the European Monetary Union (EMU) and therefore uses the euro rather than a national currency.

The syntax is as follows:

`CurrencyIsEmu` `{True|False}`

Example

```
CurrencyIsEmu True
```

CurrencySymbol

The `CurrencySymbol` option specifies the symbol used with each currency listed in the Currency Table.

The syntax is as follows:

`CurrencySymbol` *`symbol`*

where the default *label* is `*`.

Example

```
CurrencySymbol "$"
```

EmuEntryDate

The `EMUEntryDate` option specifies the date when the country became a member of the European Monetary Union (EMU) and started using the euro rather than a national currency.

The syntax is as follows:

`EmuEntryDate` *`date`*

where *date* is in `YYYYMMDD` format

Example

```
EmuEntryDate 19990101
```

EuroBaseCurrency

The `EuroBaseCurrency` option establishes the euro as the base currency in the Currency Table for this model.

The syntax is as follows:

`EuroBaseCurrency`

Example

```
EuroBaseCurrency
```

currencytableopts

Use `currencytableopts` to set the options that apply to the following MDL verbs: `CurrencyTableAdd`, `CurrencyTableMake`, `CurrencyTableUpdate`, and `CurrencyTableDelete`.

CurrencyCountryCodeColumn

The `CurrencyCountryCodeColumn` option specifies the source column that provides the country code used in the Currency Table.

The syntax is as follows:

`CurrencyCountryCodeColumn` *`objCol`*

where *objCol* is the object name and you must specify `CurrencyCountryCodeColumn`, `CurrencyLabelColumn`, or both.

Example

```
CurrencyCountryCodeColumn "Country_Code"
```

CurrencyDateColumn

The `CurrencyDateColumn` option specifies the source column that provides the effective date for the conversion rate. It is a mandatory parameter when converting currencies based on information in the Currency Table.

The syntax is as follows:

`CurrencyDateColumn` *`objCol`*

where *objCol* is the object name.

Example

```
CurrencyDateColumn "Currency Date"
```

CurrencyLabelColumn

The `CurrencyLabelColumn` option specifies the source column that provides the currency label used in the Currency Table.

The syntax is as follows:

`CurrencyLabelColumn` *`objCol`*

where *objCol* is the object name and you must specify `CurrencyCountryCodeColumn`, `CurrencyLabelColumn`, or both.

Example

```
CurrencyLabelColumn "Currency_Label"
```

CurrencyRateColumn

The `CurrencyRateColumn` option specifies the source column that provides the conversion rates. It is a mandatory parameter when converting currencies based on information in the Currency Table.

The syntax is as follows:

CurrencyRateColumn *objCol*

where *objCol* is the object name.

Example

```
CurrencyRateColumn "Conversion_Rate"
```

CurrencyTableType

The `CurrencyTableType` option specifies the type of Currency Table used in the model.

The syntax is as follows:

CurrencyTableType*type*

where *type* is one of `BaseTable`, `EuroTable`, or `OtherTable`.

Example

```
CurrencyTableType BaseTable
```

deletionopts

Use `deletionopts` to set the options that apply to the MDL verb `DeletionListUpdate`.

CalcDef

The `CalcDef` option specifies a calculation definition used in the dimension.

The syntax is as follows:

CalcDef *objDimCalcDef*

where *objDimCalcDef* is the object identifier.

Example

```
CalcDef 6375
```

Column

The `Column` option specifies a column used in the dimension.

The syntax is as follows:

Column *objCol*

where *objCol* is the object identifier.

Example

```
Column 1092
```

Cube

The `Cube` option specifies a cube used in the model.

The syntax is as follows:

Cube *objCub*

where *objCub* is the object identifier.

Example

```
Cube 1601
```

DataSource

The `DataSource` option specifies a data source used for the model.

The syntax is as follows:

DataSource *objDataSource*

where *objDataSource* is the object identifier.

Example

```
DataSource 4213
```

Dimension

The `Dimension` option specifies a dimension used in the model.

The syntax is as follows:

Dimension *objDim*

where *objDim* is the object identifier.

Example

```
Dimension 1844
```

Levels

The `Level` option specifies a level in a dimension used in the model.

The syntax is as follows:

Levels *objLev*

where *objLev* is the object identifier.

Example

```
Levels 1385
```

Measure

The `Measure` option specifies a measure used in the model.

The syntax is as follows:

Measure *objMeasure*

where *objMeasure* is the object identifier.

Example

```
Measure 1500
```

Signon

The `Signon` option specifies a signon used in the model.

The syntax is as follows:

`Signon` *`objSignon`*

where *objSignon* is the object identifier.

Example

```
Signon 9876
```

View

The `View` option specifies a dimension or security view used in the model.

For more information about custom views based on IBM Cognos 8 security objects, see the Security chapter of the Transformer *User Guide*.

The syntax is as follows:

`View` *`objView`*

where *objView* is the object identifier.

Example

```
View 1092
```

dimopts

The `dimopts` options apply to the following MDL verbs: `DimAdd`, `DimMake`, and `DimUpdate`.

Association

The `Association` option specifies the associated column from the data source.

The syntax is as follows:

`Association` *`objCol`*

where *objCol* is the object name.

Example

```
Association "Time Category Code"
```

DaysInWeek

The `DaysInWeek` option specifies each day in the week as a unique, but non-sequential number.

To specify the number of days in each week, add the appropriate day codes. For example, a Monday-to-Friday week is represented by `(2+4+8+16+32) = 62`. A seven-day Sunday-to-Saturday week (the default) is represented by the number `127`.

The syntax is as follows:

`DaysInWeek` *`bignum`*

where *bignum* is based the sum of the bit representations for the days in the week, as follows:

- Sunday = 1
- Monday = 2
- Tuesday = 4
- Wednesday = 8
- Thursday = 16
- Friday = 32
- Saturday = 64

Example

```
DaysInWeek 62
```

DimDefaultCategory

The `DimDefaultCategory` option specifies a new default category, at a lower level in the cube, where a scenario dimension opens, to show report users meaningful data values.

The syntax is as follows:

`DimDefaultCategory` *`objDimDefaultCategory`*

where *objobjDimDefaultCategory* is the object identifier.

Example

```
DimUpdate Dimension 185 "Products" DimType Regular NewCatsLock False
ExcludeAutoPartitioning False DimDefaultCategory 293
```

DimInfo

The `DimInfo` option provides additional information about the dimension.

The syntax is as follows:

`DimInfo` *`string`* [
`string``...]`

where each string may be up to 256 characters and the total description may be up to 4,095 characters.

Example

```
DimInfo "Products carried by the Great Outdoors Company."
```

DimType

The `DimType` option specifies the dimension type. Note that time dimensions are defined as a specific type (`Date`) but scenario dimensions are not.

The syntax is as follows:

`DimType` `{Regular|Date|NonStandard}`

where `NonStandard` is only used for time dimensions in which the associated data source has `PreSummarized` set to `True`, and the `Association` option is either blank or not set.

Example

```
DimType Date
```

EarliestDate

The `EarliestDate` option specifies the earliest valid date for the model.

The syntax is as follows:

`EarliestDate` *`date`*

where *date* is in the format `YYYYMMDD`. The default is `19000101`.

Example

```
EarliestDate 19000101
```

ExcludeAutoPartitioning

The `ExcludeAutoPartitioning` option specifies whether auto-partitioning will be used as a model optimization strategy.

The syntax is as follows:

`ExcludeAutoPartitioning` `{True|False}`

where the default is False.

Example

```
ExcludeAutoPartitioning True
```

LatestDate

The `LatestDate` option specifies the latest valid date for the model.

The syntax is as follows:

`LatestDate` *`date`*

where *date* is in the format `YYYYMMDD`. The default is `99991231`.

Example

```
LatestDate 99991231
```

ManualPeriods

The `ManualPeriods` option specifies whether the current time period will be manually or automatically set.

The syntax is as follows:

`ManualPeriods` `{True|False}`

where the default is `False`, which is equivalent to selecting the **Automatically Set Current Time Period** option on the Windows interface. `True` means that it the current time period will be manually set.

Example

```
ManualPeriods False
```

NewCatsLock

The `NewCatsLock` option specifies whether new categories will be automatically created in the dimension as they are encountered in the data source.

The syntax is as follows:

`NewCatsLock` `{True|False}`

where the default is `False`. `True` means that the dimension is locked.

Example

```
NewCatsLock False
```

filteropts

Use `filteropts` to set the options that apply to the following MDL verbs: `CognosPackageFilterAdd`, `CognosPackageFilterDelete` and `CognosPackageFilterUpdate`.

CognosPackageFilterRef

The `CognosPackageFilterRef` option specifies a valid filter reference in an IBM Cognos 8 package or report data source.

`CognosPackageFilterRef` *`filter reference`*

Example

```
CognosPackageFilterRef "[gosales_goretailers].[Asia Pacific]"
```

DATASOURCE

The `DATASOURCE` option specifies the ID of the IBM Cognos 8 package or report data source.

`DATASOURCE` *`datasource ID`*

Example

```
DATASOURCE 103
```

CognosPackageFilterDelete

The `CognosPackageFilterDelete` option is used to delete filters from the MDL script.

The syntax is as follows:

`CognosPackageFilterDelete` *`filter_name`*

Example

```
CognosPackageFilterDelete "Tents1"
```

CognosPackageFilterUpdate

The `CognosPackageFilterUpdate` option is used to update filters that are part of the MDL script.

The syntax is as follows:

```
CognosPackageFilterUpdate"
filter_name" DATASOURCE query_id Cognos PackageFilterRef "
package_reference_name"
```

where *query_id* is the object id of the query that is based on the package and *package_reference_name* is the name of the package followed by the name of the filter. The package that is in the package_reference_name must be in the data source that is being queried.

Example

```
CognosPackageFilterUpdate "Tents2" DATASOURCE 105 CognosPackageFilterRef "
[oracle_gosales].[Tents]"
```

ExpMark

The `ExpMark` option specifies the start of the SQL string used to query the column in the data source. The end of the string is denoted by `ExpMarkEnd`.

The syntax is as follows:

```
ExpMark SQLstring ExpMarkEnd
```

FilterDescription

The `FilterDescription` option adds a description to explain the filter used when querying a column in the data source.

The syntax is as follows:

```
Description string  [
string...]
```

where each string may be up to 256 characters and the total description may be up to 4,095 characters.

Example

```
Description "Filters out managers located in North America."
```

levelopts

Use `levelopts` to set the options that apply to the following MDL verbs: `LevelAdd`, `LevelMake`, and `LevelUpdate`.

Association

The `Association` option defines the connection between a level and columns in a data source.

The syntax is as follows:

Association *objCol*

where *objCol* is the object name.

Examples

```
Association "Time Category Code"
```

Blanks

The `Blanks` option indicates how missing values are represented within that level.

For more information about how null and missing values are treated in the supported IBM Cognos 8 reporting applications, see the Measures chapter of the Transformer *User Guide*.

The syntax is as follows:

Blanks *string*

where "N/A" represents missing values, or those which are "Not Available".

Example

```
Blanks "N/A"
```

CategoryCode

The `CategoryCode` option specifies the code used to uniquely identify a category at that level.

The syntax is as follows:

CategoryCode *objCol*

where *objCol* is the object name.

Example

```
CategoryCode "Product Line"
```

CatLabFormat

The `CatLabFormat` option specifies the format used for the date category labels at that level.

The syntax is as follows:

CatLabFormat *string*

where *string* is the desired format.

Example

The following format string produces the label 2006 Q3

```
CatLabFormat 'YYYY "Q"Q'
```

DateFunction

The `DateFunction` option specifies the time period calculation used to generate that level in the time dimension.

The syntax is as follows:

DateFunction *datefunction*

where *datefunction* is one of `None`, `Year`, `LunarYear`, `Quarter`, `LunarQuarter`, `Month`, `LunarMonth`, `LunarMonth445`, `LunarMonth454`, `LunarMonth544`, `Week`, or `Day`

Example

The following date function generates lunar months, where each 13-week lunar quarter consists of a 4-week month, a 4-week month, and a 5-week month:

```
CDateFunction LunarMonth445
```

Description

The `Description` option adds a description of the source column associated with the level.

The syntax is as follows:

`Description` *`objCol`*

where *objCol* is the object name of a column.

Example

```
Description "Product types"
```

Filtered

The `Filtered` option specifies whether the categories in the level are excluded (hidden).

The syntax is as follows:

`Filtered` `{True|False}`

Example

```
Filtered True
```

Format

The `Format` option specifies the date format used for the time categories in the level.

The syntax is as follows:

`Format` *`string`*

where *string* is a recognized date format code, such as `YMD`.

Example

```
Format "YMD"
```

Generate

The `Generate` option specifies the generation date to use when determining which categories to generate.

The syntax is as follows:

`Generate` *`generatedate`*

where *generatedate* is one of `Default`, `All`, or `Need`. On the Windows interface, selecting the check box generates all categories in the period. Clearing the check box is equivalent to the `Need` setting.

Example

```
Generate All
```

Inclusion

The `Inclusion` option specifies how the categories in a level appear in the model and in the resulting cubes and reports.

The syntax is as follows:

`Inclusion`*`inclusion`*

where *inclusion* is one of `Default`, `Generate`, `Suppress`, `Retain`, or `Filtered`.

On the Windows interface, equivalents appear on the **Inclusion** or **Diagram** menus. Note that `Generate` equates to **Always Include**, `Retain` equates to **Include When Needed** or **Suppress Blank Categories**, and `Filtered` equates to **Exclude**.

Example

```
Inclusion Suppress
```

LevelInfo

The `LevelInfo` option adds a description for the level.

The syntax is as follows:

`LevelInfo` *`string`* [
`string``...]`

where each string may be up to 256 characters and the total description may be up to 4,095 characters.

Example

```
LevelInfo "The reseller type influences the amount of sales."
```

Label

The `Label` option specifies a label for the level.

The syntax is as follows:

`Label` *`objCol`*

where *objCol* is the object name.

Example

```
Label "Customer Name"
```

NewCatsLock

The `NewCatsLock` option specifies whether new categories will be automatically created in the level as they are encountered in the data source.

The syntax is as follows:

`NewCatsLock {True|False}`

where the default is `False`. `True` means that the dimension is locked.

Example

```
NewCatsLock False
```

OrderBy

The `OrderBy` option specifies the value to use as the basis for ordering categories, within the specified drill category.

The syntax is as follows:

OrderBy Drill *objDrillCat* **Value** *string*

where *objDrillCat* can be the object name, object identifier, or both; and *string* is what appears in the **Order Value** box on the Windows interface.

Example

```
OrderBy Drill 6881 Value "200712"
```

SortOrder

The `SortOrder` option specifies the type of data used as the basis for ordering the categories, such as text. It is followed by the `SortAs` option, which determines the sorting sequence, which can be in either ascending or descending order.

If the data is alphanumeric, use the `Character` storage type. If the data is purely numeric, we recommend that you use the `Float64` storage type.

The syntax is as follows:

SortOrder Drill *objDrillCat*
Column *objCol*
SortOrder *storagetype*
SortAs {Ascending|Descending}

where *objDrillCat* and *objCol* can be the object name, object identifier, or both; and *storagetype* is one of `Default`, `Int8`, `Int16`, `Int32`, `Float64`, `Text`, `Character`, or `Longtext`.

Example

```
SortOrder Character
```

SortAs

The `SortAs` option determines the sorting sequence, which can be in either ascending or descending order. It follows the `SortOrder` option, which specifies the type of data used as the basis for sorting.

If the data is alphanumeric, use the `Character` storage type. If the data is purely numeric, we recommend that you use the `Float64` storage type.

The syntax is as follows:

SortOrder Drill *objDrillCat*
Column *objCol*
SortOrder *storagetype*
SortAs {Ascending|Descending}

where *objDrillCat* and *objCol* can be the object name, object identifier, or both; and *storagetype* is one of `Default`, `Int8`, `Int16`, `Int32`, `Float64`, `Text`, `Character`, or `Longtext`.

Example

```
SortAs Ascending
```

Partition

The `Partition` option specifies the partition number assigned to that level.

The syntax is as follows:

`Partition` *`uns`*

where *uns* is an unsigned number from `0` to `15`.

Example

```
Partition 3
```

RefreshLabel

The `RefreshLabel` option specifies whether to refresh the label for the level.

The syntax is as follows:

`RefreshLabel` `{True|False}`

Example

```
RefreshLabel True
```

RefreshDescription

The `RefreshDescription` option specifies whether to refresh the description for the level.

The syntax is as follows:

`RefreshDescription` `{True|False}`

Example

```
RefreshDescription False
```

RefreshShortName

The `RefreshShortName` option specifies whether to refresh the short name for the level.

The syntax is as follows:

`RefreshShortName` `{True|False}`

Example

```
RefreshShortName False
```

Share

The `Share` option specifies the ancestor level with which the current level has a shared, or proportional, relationship.

The `Share` function computes the share that an item is, relative to its parent. For more information about this Transformer-specific function, see the Transformer *User Guide*.

The syntax is as follows:

Share *objLevel*

where *objLevel* can be the object name or object identifier. *objLevel* must be an ancestor in the same dimension.

Example

```
Share "Product Line"
```

ShortName

The `ShortName` option specifies the short name that identifies the level.

The syntax is as follows:

ShortName *objCol*

where *objCol* is the object name.

Example

```
ShortName "CUST_SHRTN"
```

Source

The `Source` option specifies the data column that serves as the source for this level.

The syntax is as follows:

Source *objCol*

where *objCol* is the object name.

Example

```
Source "Customer Number"
```

Suppressed

The `Suppressed` option specifies whether data is to be suppressed (hidden) for this level.

The syntax is as follows:

Suppressed {True|False}

Example

```
Suppressed True
```

TimeRank

The `TimeRank` option specifies the relative ranking of user-defined levels in the time dimension.

Normally this value is automatically set by Transformer to remove ambiguities that can arise, for example, in relative time calculations. It is the value relative to other date value ranks rather than

the absolute value, that is important. The Windows interface equivalent is the **Time Level Ranking** field.

The syntax is as follows:

TimeRank *uns*

where *uns* is an unsigned number representing the time-level ranking.

Example

```
TimeRank 10
```

UniqueCategories

The `UniqueCategories` option specifies whether the categories in this level are unique.

The syntax is as follows:

UniqueCategories `{True|False}`

Example

```
UniqueCategories True
```

UniqueMove

The `UniqueMove` option specifies whether the moved categories are unique.

The syntax is as follows:

UniqueMove `{True|False}`

Example

```
UniqueMove True
```

meaopts

Use `meaopts` to set the options that apply to the following MDL verbs: `MeasureAdd`, `MeasureMake`, and `MeasureUpdate`.

ActivityMeasure

The `ActivityMeasure` option specifies the criteria used to determine the category count.

By default, the non-zero-value categories associated with all measures in the model are counted.

To create a Category Count, you can also use the `Dimension` or `Rollup` options. For more information about these alternatives, see "Dimension" (p. 247) and "Rollup" (p. 251).

The syntax is as follows:

ActivityMeasure *objMeasure*

where *objMeasure* is the object name. The default, when no ActivityMeasure is specified but one or more measures exist for a category, is to return a count.

Example

The scope of this category count is any existing measure:

```
MeasureMake 159 "Revenue" Levels 139 RollupCategoryCount Storage DefaultScale
0 Decimals 0 Sign False IsCurrency False DrillThrough False EndList
```

The following category count is for the specific measure referenced using `ActivityMeasure`:

```
Measure 111 "Cost" Levels 117 ActivityMeasure"Sales" Rollup CategoryCount
Storage Default Scale 0 Decimals 0 Sign False IsCurrency False DrillThrough
False EndList
```

Allocation

The `Allocation` option specifies the type of allocation and a measure to which it applies.

The syntax is as follows:

Allocation*allocation* {
AllocationMeasure *objMeasure*}

where *allocation* is one of `Default`, `None`, `Constant`, or `Allocate`. *objMeasure* specifies the measure and is the object name or object identifier.

Example

```
Allocation None
```

Association

The `Association` option specifies the source column associated with this measure.

The syntax is as follows:

Association *objCol*

where *objCol* is the column name.

Example

```
Association "Cost"
```

Calc

The `Calc` option specifies a calculation and the measure(s) to which it applies.

The syntax is as follows:

Calc *keywords objects*

where *calc* is a mixture of *objects* and *keywords*.

objects consist of the object name, optionally followed by the at sign (`@`) and the object identifier, such as `"Revenue@259"`. If object identifiers are suppressed, *objects* are identified by their object names only.

The measure calculation keywords supported by Transformer are as follows:

Calculation keyword	Type	Description
`ISNULL`	N/A	zero

Calculation keyword	Type	Description
IF	N/A	conditional IF expression
THEN	N/A	conditional THEN expression
ELSE	N/A	conditional ELSE expression
AND	N/A	both
OR	N/A	either
NULL	N/A	does not exist
!=	N/A	not equal to
=	N/A	equal to
<	N/A	less than
<=	N/A	less than or equal to
>	N/A	greater than
>=	N/A	greater than or equal to
/	Number	divided by
-	Date or Number	minus
*	Number	multiplied by
+	Character, Date, or Number	plus
^	Number	Returns a number raised to the power of a second number.
Absolute	Number	Converts numbers to their unsigned value.
Average	Number	Creates an arithmetic mean.
-	Date or Number	minus
*	Number	multiplied by

Calculation keyword	Type	Description
+	Character, Date, or Number	plus
Max	Number	maximum instance
Min	Number	minimum instance
Percent	Number	Computes the percentage.

Example

The following example represents an if-then-else calculation definition for "Revenue":

```
Measure 173 "Revenue" calc IF (ISNULL("Cost@177"+"Quantity@181")) THEN
("Cost@177") ELSE (("Quantity@181")
```

Decimals

The Decimals option specifies the number of decimal spaces in the source data.

You should set this option to the actual value in the data source.

You can also set this option on the Windows interface by using the **Decimal Places** box on the **Format** tab of the Measure property sheet.

The syntax is as follows:

Decimals *uns*

Example

```
Decimals 2
```

Dimension

The Dimension option specifies the drill category, levels, and/or dimension to which the measure applies.

If the measure is a Category Count, you can also use the ActivityMeasure or Rollup options. For more information about these alternatives, see "ActivityMeasure " (p. 244) and "Rollup" (p. 251).

The syntax is as follows:

Dimension *objDim*
Drill *objDrillCat*
Levels *objLevel*

where *objDim*, *objDrillCat*, and *objLevel* are the relevant object identifiers.

Example

```
Dimension "Products" Levels "Product ID"
```

DrillThrough

The `DrillThrough` option specifies whether drill-through actions are enabled and lists the supported target reports, such as the relevant .imr file for Impromptu Query Definition (IQD) data sources.

The syntax is as follows:

```
DrillThrough {True|False} [
 string] [string...]
EndList
```

where *string* is a report name with optional path.

Example

In this example, the drill-through operation is not selected:

```
DrillThrough False EndList
```

In this example, the drill-through operation is enabled to the specified .imr report:

```
DrillThrough True "c:installation_directory\bsc_msrs.imr"
EndList
```

DuplicateRollup

The `DuplicateRollup` option specifies how records with duplicate values will be rolled up or summarized.

The syntax is as follows:

DuplicateRollup*duplicatesrollup*

where *duplicatesrollup* is one of `Default`, `Sum`, `Minimum`, `Maximum`, `Average`, `First`, or `Last`.

Example

In this example, the rollup function is explicitly set to add (sum) all duplicate records:

```
DuplicateRollup Sum
```

DuplicateWeight

The `DuplicateWeight` option, along with the time state, specifies the weighting factors and functions used to obtain the weighted average of duplicate records for a measure.

The syntax is as follows:

DuplicateWeight *objMeasure*

where *objMeasure* is the object name.

Example

In this example, the **Revenue** measure is the weighting factor to use when determining the average rollup value of a set of duplicate records:

```
DuplicateWeight "Revenue"
```

Exclude

The `Exclude` option, followed by a list of excluded objects, restricts the scope of drill-through options so that only the non-excluded level(s) can serve as a starting point for drill-through opera-

tions. Use with the `DrillThrough` option to ensure that report users navigate directly to the relevant portion of the cube before they begin a drill-through operation.

For each or cube or measure for which you want to restrict drill-through by level, you must specify the relevant dimension and levels, and any drill-through targets to which the restriction applies.

You must use MDL if you want to exclude levels in a sub-dimension, because these are not displayed on the Dimension map of the Windows interface.

The syntax is as follows:

Exclude [**Dimension** *objDim*] [**Drill** *objDrill*] **Levels** *objLevel*

where *objDim*, *objLevels*, and *objDrill* can be the unique object identifiers, names, or both.

Example

In this example, the **State** and **Market** levels in the **Region** and **Sales** dimensions are excluded for the specified cube drill-through operations:

```
DrillThrough True ... "C:\path_to_cube.mdc" "Description
text" Exclude Dimension "Region" Exclude Levels 219 "State" Exclude Dimension
"Sales" Exclude Levels "Market" EndExclude
"C:\path_to_cube2.mdc" "Description text2" EndList
```

Format

The `Format` option specifies the measure format and the number of decimal places to use.

The syntax is as follows:

Format *string*

where *string* is the format, followed by a tilde (~), and the number of decimal places.

Example

In this example, the measure format is `#,##0` and the number of decimal places is set to `2`:

```
Format "#,##0~2"
```

IgnoreMissingValue

The `IgnoreMissingValue` option specifies whether to ignore null and missing (`NA`) values when performing certain kinds of time-state rollup (Average and Weighted Average). If you do not specify the option when you create, make, or update a measure, the default setting (include these values) applies.

You cannot use the `IgnoreMissingValue` option with time-state rollups of type `None`, `First Period`, `Last Period`, or `Current Period`. Null and missing values are always included when aggregating or calculating these kinds of rollup.

Null and missing data values are always excluded from `Min` and `Max` calculations for rollups, whether they are set by Transformer to display as `0` or `na` (the `NA` setting).

If you do not specify this option, its value is assumed to be `False`.

For more information about specifying measures, see "MeasureMake" (p. 164).

The syntax is as follows:

TimeStateRollup*type*

where *type* is one of `Average` or `Weighted Average`.

Example

This example uses structured MDL to ignore missing values for a `Weighted Average` time state rollup:

```
Measure 173 "Cost" TimeStateRollup Average TimeStateWeight "Outlet"
IgnoreMissingValue True Storage Default
OutPutScale 0 Decimals 0 ReverseSign False IsCurrency False IsFolder False
WeightId 243 DrillThrough False EndList Associations 175 "Revenue"
AssociationType Type_Query
AssociationRole Role_Source AssociationReferenced "Revenue"
```

This example uses verb MDL to ignore missing values for an `Average` time state rollup:

```
MeasureCreate 173 "Profit" TimeStateRollup Average IgnoreMissingValue True
Storage Default
OutPutScale 0 Decimals 0 ReverseSign False IsCurrency False IsFolder False
WeightId 243 DrillThrough False EndList Associations 175 "Revenue"
AssociationType Type_Query AssociationRole Role_Source
AssociationReferenced "Revenue"
```

IsCurrency

The `IsCurrency` option specifies whether the measure is a currency value.

The syntax is as follows:

IsCurrency {True|False}

Example

```
IsCurrency True
```

IsFolder

The `IsFolder` option specifies whether the measure is a folder that groups several measures.

The syntax is as follows:

IsFolder {True|False}

Example

```
IsFolder True
```

Label

The `Label` option specifies the label used for the measure.

The syntax is as follows:

Label *string*

Example

```
Label "Qty"
```

MeasureInfo

The `MeasureInfo` option provides a description of the measure.

The syntax is as follows:

`MeasureInfo` *`string`* `[`
`string...``]`

where each string may be up to 256 characters and the total description may be up to 4,095 characters.

Example

```
MeasureInfo "Cost to purchase products"
```

Missing

The `Missing` option specifies how records with missing or null values are to be represented in the reporting components.

The syntax is as follows:

`Missing` `[`*`missingvalue`*`]`

where *missingvalue* is one of `Default`, `Zero`, or `NA` (reports show `na`).

Example

```
Missing Zero
```

Rollup

The `Rollup` option specifies how measure values are summarized or rolled up, across each reporting time period.

The `External` rollup option indicates that the measure was rolled-up before it was imported into Transformer, and so retains the summarization method established in the data source.

For a Category Count measure, you can also use the `ActivityMeasure` or `Dimension` options. For more information about these alternatives, see "ActivityMeasure " (p. 244) and "Dimension" (p. 247).

The syntax is as follows:

`Rollup` `[`*`rollup`*`]`

where *rollup* is one of `CategoryCount`, `Default`, `Sum`, `Minimum`, `Maximum`, `Average`, `Count`, `CountAll`, `Any`, `None`, or `External`.

Example

```
Rollup Average
```

Parent

The `Parent` option uniquely identifies the parent object of the measure, where the model contains a hierarchy of measures.

The syntax is as follows:

`Parent` *`objMeasure`*

where *objMeasure* is the unique identifier of the parent.

Example

```
Parent 4719
```

Scale

The `Scale` option specifies the output scale for the measure.

The syntax is as follows:

`Scale` *`uns`*

where *uns* is an unsigned number and zero (`0`) signifies no change in scale.

Example

```
Scale 0
```

ShortName

The `ShortName` option specifies the short name used to represent the measure.

The syntax is as follows:

`ShortName` *`string`*

Example

```
ShortName "QtySold"
```

Sign

The `Sign` option specifies whether the sign of a measure value is to be reversed, changing its value from a negative to a positive number, or from a positive to a negative number. Use this option to convert values into the form required by specific financial reports, such as debits and credits, or assets and liabilities.

The syntax is as follows:

`Sign` `{True|False}`

Example

```
Sign False
```

Storage

The `Storage` option specifies the data type used to store the source values for this measure.

The syntax is as follows:

`StorageType` `[`
`storagetype``]`

where *storagetype* is one of `Default`, `Int8`, `Int16`, `Int32`, `Float64`, `Text`, `Character`, or `LongText`.

Example

```
Storage Float64
```

TimeStateRollup

The `TimeStateRollup` option specifies the function used to summarize or roll up measure values for a specific time period.

Note: Time-state rollups only occur at runtime. This is why `External` can be specified as a rollup parameter, but not as a timerollup parameter.

The syntax is as follows:

`TimeStateRollup` [
`timerollup`]

where *timerollup* is one of `Default`, `Minimum`, `Maximum`, `First`, `Last`, `Current`, `Average`, or `Summary`.

Example

This example uses verb MDL to create an `Average` time state rollup for the Profit measure, and ignore any missing values:

```
MeasureCreate 173 "Profit" TimeStateRollup Average IgnoreMissingValue True
Storage Default OutPutScale 0 Decimals 0 ReverseSign False IsCurrency
False IsFolder False WeightId 243 DrillThrough False EndList Associations 175
"Revenue" AssociationType Type_Query AssociationRole Role_Source
AssociationReferenced "Revenue"
```

TimeStateWeight

The `TimeStateWeight` option specifies the object used to obtain the weighting for a time state rollup. Note that the `TimeStateRollup` option must be explicitly set to `Average`.

The syntax is as follows:

`TimeStateWeight` *`objMeasure`*

where *objMeasure* is the object name.

Example

This example uses structured MDL to create an average `Cost` measure, weighted by, or in proportion to, the measures associated with the `Outlet` object. Missing values are ignored.

```
Measure 173 "Cost" TimeStateRollup Average
TimeStateWeight "Outlet"
IgnoreMissingValue True Storage Default OutPutScale 0 Decimals 0 ReverseSign
False IsCurrency False IsFolder False WeightId 243 DrillThrough
False EndList Associations 175 "Revenue" AssociationType Type_
Query AssociationRole Role_Source AssociationReferenced "Revenue"
```

Timing

The `Timing` option specifies the timing that applies to the measure; that is, before or after roll-up.

The syntax is as follows:

`Timing` [*`measuretiming`*]

where *measuretiming* is one of `Default`, `Before_Rollup`, or `After_Rollup`.

Example

```
Timing After_Rollup
```

Weight

The `Weight` option specifies the measure to use for weighting the time-state rollup. This option requires an object name, whereas the `WeightID` option uses the object identifier. For both, the `TimeStateRollup` option must be set to `Average`.

The syntax is as follows:

`Weight` *`objMeasure`*

where objMeasure is the object name.

Example

```
Weight "Revenue"
```

WeightID

The `WeightID` option specifies the measure to use for weighting the time-state rollup. This option requires an object identifier, whereas the `Weight` option uses the object name. For both, the `TimeStateRollup` option must be set to `Average`.

The syntax is as follows:

`WeightID` *`objMeasure`*

where *objMeasure* is the object identifier.

Example

```
Weight 259
```

powercubeopts

Use `powercubeopts` to set the options that apply to the following MDL verbs: `CubeAdd`, `CubeMake`, `CubeUpdate`, `CubeGroupAdd`, `CubeGroupMake`, `CubeGroupUpdate`, `CubeGroupCubeAdd`, `CubeGroupCubeMake`, and `CubeGroupCubeUpdate`.

BlockParentTotals

The `BlockParentTotals` option specifies whether the total values for parent categories are included or hidden for the specified cube or cube group. Use this option to display the word "denied" in the reporting component, rather than a total value that would represent an inaccurate rollup of only non-excluded categories.The Windows equivalent is to select the **Block Total Values for Parent Categories with Excluded Children** option when categories have been excluded from a custom view. For this to work properly, you must ensure that the appropriate security objects have been enabled.

The syntax is as follows:

`BlockParentTotals` `{True|False}`

where `False` is the default.

Example

```
BlockParentTotals True
```

Caching

The `Caching` option specifies whether crosstab caching is enabled. The option sets a flag in the .mdc cube file.

The syntax is as follows:

`Caching` `{True|False}`

Example

```
Caching True
```

Compress

The `Compress` option specifies whether the cube is compressed. This option may appear in legacy cubes but is not supported as an optimization technique for IBM Cognos 8.

The syntax is as follows:

`Compress` `{True|False}`

Example

```
Compress False
```

Consolidate

The `Consolidate` option specifies the record consolidation method used for the cube or cube group.

The syntax is as follows:

`Consolidate`*`consolidate`*

where *consolidate* is one of `Default`, `Yes`, `No`, or `PreSorted`.

Example

```
Consolidate Yes
```

ConsolidatedRecords

The `ConsolidatedRecords` option specifies the number of records consolidated during each partitioning pass.

The syntax is as follows:

`ConsolidatedRecords` *`bignum`*

where the default is `10,000,000` records.

Example

```
ConsolidatedRecords 10,000,000
```

CubeCreation

The `CubeCreation` option enables or disables the creation of specific cubes in the model.

The syntax is as follows:

CubeCreation*makecube*

where *makecube* is one of `Default`, `Implicit`, `On`, or `Off`.

Example

```
CubeCreation On
```

CubeCycle

The `CubeCycle` option is related to internal Transformer processing and should not be set.

The syntax is as follows:

CubeCycle *bignum*

Example

```
CubeCycle 604
```

CubeStamp

The `CubeStamp` option is related to internal Transformer processing and should not be set.

The syntax is as follows:

CubeStamp *bignum*

Example

```
CubeStamp 861985194
```

CubeUpdateLock

The `CubeUpdateLock` option specifies whether no cubes or all cubes are locked for update.

The syntax is as follows:

CubeUpdateLock `{None|All}`

Example

```
CubeUpdateLock None
```

Database

The `Database` option specifies the database providing the source data for a cube or cube group. For example, you can use this option to identify the source for an IQD signon.

The syntax is as follows:

DataBase *objSignon*

where *objSignon* can be the object name, object identifier, or both.

Example

```
Database 6317
```

DatabaseInfo

The `DatabaseInfo` option provides information about the type of database used with the cube or cube group.

The syntax is as follows:

DatabaseInfo *string*

where the database type and settings are separated by semi-colons.

Example

```
DatabaseInfo "Local;;"
```

DataSource

The `DataSource` option specifies a main Impromptu Query Definition (IQD) data source and one or more alternate data sources, with their associated options.

For more information on IBM Cognos 8 data sources and options, see "appqueryopts" (p. 197).

The syntax is as follows:

DataSource *objDataSource*
AlternateSource *filename* [
altdatasourceopts]

where *objDataSource* is the object name and *altdatasourceopts* is one or more of the following:

- Isolation*number*
- DmsSignon*objSignon*
- IMRName*string*
- SourceSignonList *signonsignons*...] EndList
- SQL*string*
- Stamp*number*

Example

The following example shows the SQL for an Impromptu Query Definition (IQD) data source:

```
DataSourceMake 103 "Report1" Separator " " SourceType DataSource DecimalSep "
" Thousandsep " " Columns True Timing PopYesCreateDefault Source "C:\Report1.
iqd" SQL 'select T1."PROD_TYPE" as c1,' 'T1."PROD_LINE" as c2, T1."PRODUCT"'
 'as c3, T1."PROD_COST" as c4,' 'T2."ORDER_DT" as c5 from
"ORDRDETL" T3,' '"PRODUCT" T1,"ORDER" T2' 'where (T3."PROD_NO" = T1."PROD_
NO")' 'and (T2."ORDER_NO" = T3."ORDER_NO") and' '(T1."PROD_TYPE" =
"Outdoor Products) " Isolation 0 SourceSignonList 105 1105 EndList ImrName "C:
\Report1.imr" Stamp 930750524  Speed False SetCurrent True ServerSource
False Presummarized False
```

DeployLocations

The `DeployLocations` option specifies a list of deployment locations for the PowerCube.

The syntax is as follows:

DeployLocations*value* Endlist

Example

```
DeployLocations "C:\Cubes" Endlist
```

DeployType

The `DeployType` option specifies the type of PowerCube deployment strategy to use for the PowerCube.

The syntax is as follows:

DeployType*deploytype*

where *deploytype* is one of DeployNone, DeploySwapTogether, or DeploySwapSingle (not available in this release).

Example

```
DeployType DeploySwapTogether
```

DetailLevel

The `DetailLevel` option identifies the level that contains the categories on which individual cubes in a cube group are based. Use to create a cube for each category in the level. Duplicate level names in the list are qualified by the name of the owning category, as in Brand (OF Tent).

The syntax is as follows:

DetailLevel *objLevel*

where *objLevel* can be the object name.

Example

```
DetailLevel "Product Type"
```

DrillThrough

The `DrillThrough` option specifies whether drill- through action is enabled for the cube or cube group, and lists the target report objects.

The syntax is as follows:

DrillThrough [True|False]
reportnames **EndList**

where *reportnames* are the reports, as listed in the **Custom Reports** and **Report Description** boxes on the Windows interface.

Example

In this example, drill through is enabled for the specified .mdc cube file:

```
DrillThrough True "c:installation_directory\all_regions.mdc"
EndList
```

Exclude

The `Exclude` option is used, followed by a list of excluded objects, to restrict the scope of drill-through activity to targets related to a specific non-excluded drill category or dimension level. By

so doing, the modeler can ensure that users navigate directly to the relevant portion of the cube before they begin their drill-through operation.

For each or cube or measure for which to want to restrict drill-through activity by level, you must specify the relevant dimension and levels, and any drill-through targets to which the restriction applies.

You must use MDL if you want to exclude levels in a sub-dimension, because these are not displayed on the **Dimension Map** of the Windows interface.

The syntax is as follows:

Exclude [**Dimension** *objDim*] [**Drill** *objDrill*] **Levels** *objLevel*

where *objDim*, *objLevels*, and *objDrill* can be the unique object identifiers, names, or both.

Example

In this example, the State and Market levels in the Region and Sales dimensions are excluded for the specified cube drill-through operations:

```
DrillThrough True ...  "C:\path_to_cube.mdc" "Description text" Exclude
Dimension "Region" Exclude Levels 219 "State"  Exclude Dimension "Sales"
Exclude Levels "Market" EndExclude "C:\path_to_cube2.mdc" "Description
text2" EndList
```

IncrementalUpdate

The `IncrementalUpdate` option specifies whether incremental updates are enabled for this cube or cube group.

An object identifier is automatically assigned when the cube is incrementally updated.

The syntax is as follows:

IncrementalUpdate {True|False}

Example

```
IncrementalUpdate True
```

Information

The `Information` option provides a description of the cube or cube group.

The syntax is as follows:

Information *string* [
string...]

where each string may be up to 256 characters and the total description may be up to 4,095 characters.

Example

```
Information "GO Sales Sample PowerCube."
```

MDCFile

The `MDCFile` option specifies the path and file name used for the .mdc cube file.

The syntax is as follows:

MDCFile *string*

Example

```
MDCFile "c:\Temp\outdoors.mdc"
```

MeasureName

The `MeasureName` option specifies the name used for a cube measure or group of measures.

The syntax is as follows:

MeasureName *string*

Example

```
MeasureName "Basic Measures"
```

Optimize

The `Optimize` option specifies the optimization strategy applied to the cube or cube group. For more information about optimizing cube build times, see the Transformer *User Guide*.

The syntax is as follows:

Optimize*cubeoptimize*

where *cubeoptimize* is one of `Default`, `Categories`, `Datapasses`, `DirectCreate`, `PriorMarking`, or `AutoPartition`.

Example

```
Optimize Categories
```

PartitionSize

The `PartitionSize` option specifies the number of records included in each partition.

The syntax is as follows:

PartitionSize *bignum*

Example

```
PartitionSize 500,000
```

PassesNumber

The `PassesNumber` option specifies the number of partitioning passes to run.

The syntax is as follows:

PassesNumber *bignum*

where *bignum* is any number; the default is `5`.

Example

```
PassesNumber 4
```

Password

The `Password` option specifies an unencrypted cube password.

The syntax is as follows:

`Password` *`string`*

Example

```
Password "mypasswd"
```

PublishAllowAccessToSuppressionOptions

The `PublishAllowAccessToSuppressionOptions` option specifies whether the package published from Transformer allows the user to access null suppression options in the Web studios.

The syntax is as follows:

`PublishAllowAccessToSuppressionOptions` `{True|False}`

Example

```
PublishAllowAccessToSuppressionOptions False
```

PublishAllowMultiEdgeSuppression

The `PublishAllowMultiEdgeSuppression` option specifies whether the package published from Transformer allows the user to enable multi-edge suppression in the Web studios.

The syntax is as follows:

`PublishAllowMultiEdgeSuppression` `{True|False}`

Example

```
PublishAllowMultiEdgeSuppression False
```

PublishAllowNullSuppression

The `PublishAllowNullSuppression` option specifies whether the package published from Transformer allows the user to control null suppression in the Web studios.

The syntax is as follows:

`PublishAllowNullSuppression` `{True|False}`

Example

```
PublishAllowNullSuppression False
```

SegmenterDimension

The `SegmenterDimension` option specifies the name of the dimension on which a cube group is based or partitioned.

The syntax is as follows:

`SegmenterDimension` *`objDim`*

where *objDim* is the object name.

Example

```
SegmenterDimension "Products"
```

SegmenterLevel

The `SegmenterLevel` option specifies the name of the level on which a cube group is based or partitioned.

The syntax is as follows:

`SegmenterLevel` *`objLevel`*

where *objLevel* is the object name.

Example

```
SegmenterLevel "Product Line"
```

ServerCube

The `ServerCube` option specifies whether the cube is located on the server, which is relevant in versions of Transformer that support client/server processing.

The syntax is as follows:

`ServerCube` `{True|False}`

where True is equivalent to selecting **On the Server** in Transformer version 7.x. Client-server processing is not supported in Transformer version 8.x.

Example

```
ServerCube False
```

Status

The `Status` option specifies the processing status of the cube or cube group.

The syntax is as follows:

`Status` `[`*`cubestatus`*`]`

where *cubestatus* is one of `New`, `OK`, `Inactive`, `Warnings`, `Invalid`, `Failed`, `Busy`, or `Missing`.

Example

```
Status Inactive
```

SummaryLevel

The `SummaryLevel` option specifies the level at which category values are summarized.

The Windows interface equivalent is the **Summarize all external categories in the level** option on the **Cube Group** tab of the **Cube** property sheet.

The syntax is as follows:

`SummaryLevel` *`string`*

Example

```
SummaryLevel "Product Type"
```

UseAlternateFilename

The `UseAlternateFilename` option specifies whether to use an alternate file name for the cube.

The syntax is as follows:

`UseAlternateFilename` `{True|False}`

Example

```
UseAlternateFilename True
```

TimeBasedPartitionedCube

The `TimeBasedPartitionedCube` option specifies whether time-based partitioning is enabled for the cube.

The syntax is as follows:

`TimeBasedPartitionedCube` `{True|False}`

Example

```
TimeBasedPartitionedCube True
```

promptopts

Use `promptopts` to set the options that apply to data source prompts and the following MDL verbs: `PromptAdd`, `PromptMake`, and `PromptUpdate`.

PromptType

For data source prompts, the `PromptType` option specifies the type of prompt to be imported into the Transformer model.

The syntax is as follows:

`PromptType` `{xsdString|memberUniqueName}`

Use "xsdString" as the prompt type for simple-value prompts, multi-value prompts, and range prompts.

Examples

```
PromptType "xsdString" PromptType "memberUniqueName"
```

PromptValue

For data source prompts, the `PromptValue` option specifies the value of the prompts imported into the Transformer model.

The syntax is as follows:

`PromptValue` *`value`*

Examples

The following example specifies a prompt value of 2 for a simple-value prompt.

```
PromptValue 2
```

The following example specifies prompt values of 0, 1, and 2 for a multi-value prompt.

```
PromptValue StartList "0""1""2" EndList
```

The following example specifies prompt values of 1 through 22 for a range prompt.

```
PromptValue StartRange "1""22" EndRange
```

The following example specifies the prompt value for a Member Unique Name prompt.

```
PromptValue StartList "[ZQ1AUTCTY WORLD].[AMERICA ZQ1AUTCNT]"EndList
```

signonopts

Use `signonopts` to set the options that apply to data sources and the following MDL verbs: `SignonAdd`, `SignonMake`, and `SignonUpdate`.

PromptForPassword

For data source signons, the `PromptForPassword` option specifies whether the user is prompted to enter their signon password.

The syntax is as follows:

`PromptForPassword` `{True|False}`

Example

```
PromptForPassword True
```

UserId

The `UserId` option specifies the user ID portion of the signon.

The syntax is as follows:

`UserId` ***`"user name"`***

Example

```
UserID "corpadm"
```

password

The `password` option specifies the signon password in unencrypted form.

The syntax is as follows:

`password` ***`"password"`***

Example

```
password "my_pass"
```

AutoLogon

For Cognos 8 signons, the `AutoLogon` option specifies whether to prompt for credentials as part of the Cognos 8 signon process.

The syntax is as follows:

`AutoLogon` `{True|False}`

Example

```
AutoLogon True
```

SignonNamespace

For Cognos 8 signons, the `SignonNamespace` option specifies the Cognos 8 namespace for the signon.

The syntax is as follows:

SignonNamespace ***"namespacename"***

Example

```
SignonNamespace "Cognos"
```

SignonType

The `SignonType` option specfies whether the signon is a Cognos 8 or data source signon.

The syntax is as follows:

SignonType {Cognos|DataSource}

Example

```
SignonType Cognos
```

SignonInfo

The `SignonInfo` option provides a description of the signon.

The syntax is as follows:

SignonInfo *description* [
description...]

where each string may be up to 256 characters and the total description may be up to 4,095 characters.

Example

```
SignonInfo "The logical database name that contains the
source data for the Cognos 8 data source."
```

viewopts

Use `viewopts` to set the options that apply to the following MDL verbs: `ViewAdd`, `ViewMake`, and `ViewUpdate`. These act on dimension views and custom security views. For more information about the latter, see the Security section of the Transformer *User Guide*.

Apex

The `Apex` option specifies which category is to be apexed and become the top level. For more information, see the Transformer *User Guide*.

The syntax is as follows:

Apex *objCat*

where *objCat* specifies the category and is the object name, object identifier, or both.

Example

```
Apex "Europe"
```

Cloak

The `Cloak` option specifies which category is to be cloaked or hidden. For more information, see the Transformer *User Guide*.

The syntax is as follows:

Cloak *objCat*

where *objCat* specifies the category and is the object name, object identifier, or both.

Example

```
Cloak "Far East"
```

Filter

The `Filter` option specifies which category is to be excluded, or filtered, from the cube. For more information, see the Transformer *User Guide*.

The syntax is as follows:

Filter *objCat*

where *objCat* specifies the category and is the object name, object identifier, or both.

Example

```
Filter "Europe"
```

LevelCloak

The `LevelCloak` option specifies which level and drill category are to be cloaked or hidden. For more information, see the Transformer *User Guide*.

The syntax is as follows:

LevelCloak *objLevel* [
Drill *objDrillCat*]

where *objLevel* specifies the level and is the object name, object identifier or both. *objDrillCat* specifies the drill category and is the object name, object identifier, or both.

Example

```
LevelCloak "Customer Type"
```

LevelFilter

The `LevelFilter` option specifies which level and drill category are to be excluded, or filtered, from the cube. For more information, see the Transformer *User Guide*.

The syntax is as follows:

LevelFilter *objLevel* [
Drill *objDrillCat*]

where *objLevel* specifies the level and is the object name, object identifier or both. *objDrillCat* specifies the drill category and is the object name, object identifier, or both.

Example

```
LevelFilter "Customer Type"
```

LevelSummary

The `LevelSummary` option specifies which level and drill category are to be summarized. For more information, see the Transformer *User Guide*.

The syntax is as follows:

```
LevelSummary objLevel [
Drill objDrillCat]
```

where *objLevel* specifies the level and is the object name, object identifier or both. *objDrillCat* specifies the drill category and is the object name, object identifier, or both.

Example

```
LevelSummary "Region"
```

LevelSuppressed

The `LevelSuppressed` option specifies which level is to be suppressed. For more information, see the Transformer *User Guide*.

The syntax is as follows:

```
LevelSuppressed objLevel
```

where *objLevel* specifies the level and is the object name, object identifier, or both.

Example

```
LevelSuppressed "Region"
```

Name

The `Name` option specifies a name for the view.

The syntax is as follows:

```
Name objView
```

where *objView* is the object name.

Example

```
Name "Europe"
```

NotCloak

The `NotCloak` option specifies which category is not to be cloaked or hidden. For more information, see the Transformer *User Guide*.

The syntax is as follows:

```
NotCloak objCat
```

where *objCat* specifies the category and is the object name, object identifier, or both.

Example

```
NotCloak "Far East"
```

NotFilter

The `NotFilter` option specifies which category is not to be excluded, or filtered, from the cube. For more information, see the Transformer *User Guide*.

The syntax is as follows:

`NotFilter` *`objCat`*

where *objCat* specifies the category and is the object name, object identifier, or both.

Example

```
NotFilter "Customer Type"
```

NotSummary

The `NotSummary` option specifies which category is not to be summarized. For more information, see the Transformer *User Guide*.

The syntax is as follows:

`NotSummary` *`objCat`*

where *objCat* specifies the category and is the object name, object identifier, or both

Example

```
NotSummary "Region"
```

NotSuppressed

The `NotSuppressed` option specifies which category is not to be suppressed. For more information, see the Transformer *User Guide*.

The syntax is as follows:

`NotSuppressed` *`objCat`*

where *objCat* specifies the category and is the object name, object identifier, or both.

Example

```
NotSuppressed "Region"
```

Summary

The `Summary` option specifies which category is to be summarized. For more information, see the Transformer *User Guide*.

The syntax is as follows:

`Summary` *`objCat`*

where *objCat* specifies the category and is the object name, object identifier, or both

Example

```
Summary "Europe"
```

Suppressed

The `Suppressed` option specifies which category is to be suppressed. For more information, see the Transformer *User Guide*.

The syntax is as follows:

`Suppressed` *`objCat`*

where *objCat* specifies the category and is the object name, object identifier, or both.

Example

```
Suppressed "Far East"
```

Chapter 7: Structured MDL

There are two forms of MDL: verb MDL, which is the main focus of the *Developer Guide*, and structured MDL, which is briefly described is this section, under the following topics

Like verb MDL, the structured form of MDL can provide a complete and functionally independent definition of any Transformer model.

History of MDL

Model Definition Language (MDL) was created to store Transformer models so they could be opened in future versions of Transformer or related IBM Cognos modeling components. In its original form, MDL contained statements that defined the model in a fixed order: that is, structured MDL.

Verbs were added to enhance the functionality of the language and to support software developers by providing an independent, parallel, workable format: that is, verb MDL.

To maintain compatibility between product versions, structured MDL continues to be the default form. However, if you are building models using MDL, we recommend that you add the cogtr.ini setting `VerbOutput=1`, so that Transformer automatically generates verb output.

For more information about global preference settings, see the Transformer *User Guide*.

Comparison of Structured and Verb Keywords

The following table lists Transformer objects. Beside each object is the structured keyword that is used in MDL models to define the object, and the verbs that define and manipulate the object.

Note: Most but not all of the currently-supported verbs are included in this list.

Transformer object	Structured keyword	Verbs
Model	Name	NewModel
Data Source	DataSource, CognosSource	DataSourceAdd, DataSourceDelete, SourceListUpdate, DataSourceMake, DataSourceUpdate
Filter	Filter, FilterRef	CognosPackageFilterAdd, CognosPackageFilterMake, CognosPackageFilterDelete, CognosPackageFilterUpdate

Transformer object	Structured keyword	Verbs
Prompt	Prompt	PromptAdd, PromptDelete, PromptMake, PromptUpdate
Column	OrgName	ColumnAdd, ColumnDelete, ColumnListUpdate, ColumnMake, ColumnUpdate
Dimension	Dimension	DimAdd, DimDelete, DimMake, DimUpdate, DimensionListUpdate
Subdimension	SubDimCat	SubDimRootMake, SubDimRootUpdate
Root Category	Root	RootCatMake, RootCatUpdate
Drill Category	Drill	DrillCatMake
Level	Levels	LevelAdd, LevelDelete, LevelMake, LevelMoveAfter, LevelMoveBefore, LevelNewDrill, LevelUpdate, FilterLevel
Category	Category	CatAdd, CatDelete, CatJoin, CatMake, CatMorph, CatMoveLevel, CatMoveVertical, CatUpdate, FilterCat
Special Category	SpecialCategory	SpecialCatAdd, SpecialCatDelete, SpecialCatMake, SpecialCatUpdate
Dimension View	ViewName, DimensionView	ApexCat, ViewAdd, ViewDelete, ViewListUpdate, ViewMake, ViewUpdate, CustomViewChildListUpdate
Custom View	ViewCustomView, CustomView	CustomViewAdd, CustomViewMake, CustomViewDelete, CustomViewUpdate, CustomViewChildListUpdate, PowerCubeCustomViewListUpdate, SecurityNamespaceAdd, SecurityNamespaceDelete, SecurityNamespaceMake, SecurityNamespaceUpdate, SecurityObjectAdd, SecurityObjectDelete, SecurityObjectMake, SecurityObjectUpdate
Measure	Measure	MeasureAdd, MeasureDelete, MeasureMake, MeasureListUpdate, MeasureUpdate

Transformer object	Structured keyword	Verbs
PowerCube	Cube	CubeAdd, CubeDelete, CubeMake, CubeUpdate, PowerCubeDelete, PowerCubeListUpdate, PowerCubeSecurityListUpdate
PowerCube Group	CubeGroup	CubeGroupAdd, CubeGroupDelete, CubeGroupMake, CubeGroupUpdate, PowerCubeDelete, PowerCubeListUpdate
PowerCube Group Cube	CubeGroupCube	CubeGroupCubeAdd, CubeGroupCubeDelete, CubeGroupCubeListUpdate, CubeGroupCubeMake, CubeGroupCubeUpdate
Signon	Signon	SignonAdd, SignonDelete, SignonListUpdate, SignonMake, SignonUpdate, CognosPackageDatasourceConnectionAdd, CognosPackageDatasourceConnectionMake, CognosPackageDatasourceConnectionUpdate, CognosPackageDatasourceConnectionDelete
Dimension Calculation Definition	DimCalcDef	DimCalcDefAdd, DimCalcDefDelete, DimCalcDefMake, DimCalcDefUpdate
Currency Table	CurrencyTable	CurrencyTableAdd, CurrencyTableDelete, CurrencyTableMake, CurrencyTableUpdate
Currency	CurrencyRecord	CurrencyAdd, CurrencyDelete, CurrencyMake, CurrencyUpdate, CurrencyTableAdd
Security View	SecurityNameSpace and SecurityObject	NameSpaceMake, SecurityObjectMake

Using Structured MDL

Structured MDL requires that object definitions appear in a specific order or structure. For example, after a data source definition, all the columns in the data source must be defined. Then the associated dimensions are defined, followed by the root category, drill category, levels, and categories.

This structure provides information about the location of each object. For example, to uniquely identify a category in verb MDL, you may need to specify its parents, level, drill category, and dimension. In structured MDL, all this information is supplied by the placement of the category definition in the structured MDL file.

Verb statements can be used independently or in conjunction with structured statements, as long as the verb statements are at the end of the model. After a verb statement, no structured statements can be used.

Example

The following is an example of a model in structured MDL format:

```
Name "New Model" ModelCodePage "windows-1252" AutoAccess
False
SynchroCycle 0 SynchroStamp 0 UpdateCycle 1 ModelStamp 1189458791
ClientStamp 1189458791 ServerStamp 0 Version "8.3.790.0"
ModelCategoryOrderDefault
OrderUsePreference ModelOrderedByDefault False CognosSource 103
"GO Sales and Retailers"
SourceType Package
SourcePath "/content/package[@name='GO Sales and Retailers']"
PackageTimeStamp "/content/package[@name='GO Sales and Retailers']/model
[@name='model']"
CognosSource 11543 "QuantityReport"
SourceType Report
SourcePath "/content/package[@name='GO Sales and Retailers']/report
[@name='QuantityReport']"
PackageTimeStamp "/content/package[@name='GO Sales and Retailers']/report
[@name='QuantityReport']"
DataSource 105 "GO Sales and Retailers~1" Separator ","
SourceType CognosSourceQuery
CharacterSet Default DecimalSep "." Thousandsep "," Columns True
Timing PopYesCreateDefault
PackageReportSource 103 "GO Sales and Retailers" AutoSummary True
SetCurrent True ServerSource False Speed False Presummarized False
OrgName 107 "[gosales_goretailers].[Orders].[Order number]"
Origin Source Offset 0
Column "Order number" Storage Float64 Scale 0
Size 4 Decimals 0
InputScale 0 TimeArray Off Rollup CountAll
OrgName 109 "[gosales_goretailers].[Orders].[Retailer name]"
Origin SourceOffset 1
Column "Retailer name" Storage Text Scale 0
Size 102 Decimals 0
Class Description InputScale 0 TimeArray Off
OrgName 111 "[gosales_goretailers].[Orders].[Order date]"
Origin Source Offset 2
Column "Order date" Storage Int32 Scale 0
Size 12 Decimals 0
Class Date InputScale 0 TimeArray Off
OrgName 113 "[gosales_goretailers].[Orders].[Order method]"
Origin Source Offset 3
Column "Order method" Storage TextScale 0
Size 102 Decimals 0
Class Description InputScale 0 TimeArray Off
OrgName 115 "[gosales_goretailers].[Orders].[Order method code]"
Origin Source Offset 4
Column "Order method code" Storage Float64 Scale 0
Size 4 Decimals 0 InputScale 0
TimeArray Off Rollup CountAll
OrgName 117 "[gosales_goretailers].[Orders].[Product name]"
Origin Source Offset 5
Column "Product name" Storage Text Scale 0
Size 102 Decimals 0
Class Description InputScale 0 TimeArray Off
OrgName 119 "[gosales_goretailers].[Orders].[Quantity]"
Origin Source Offset 6
Column "Quantity" Storage Float64 Scale 0
```

```
Size 2 Decimals 0
Class Quantity InputScale 0 TimeArray Off
Rollup Sum Filter 11553 "Americas" FilterRef
"[gosales_goretailers].[Americas]"
DataSource 11545 "QuantityReport~1" Separator ","
SourceType CognosSourceQuery
CharacterSet Default DecimalSep "." Thousandsep "," Columns True
Timing PopYesCreateDefault
PackageReportSource 11543"QuantityReport" AutoSummary False SetCurrent
True ServerSource False
Speed False Presummarized False
OrgName 11547 "[Report].[Query1.0].[Product number]"
Origin Source Offset 0
Column "Product number" Storage Float64 Scale 0
Size 1 Decimals 0
Class Quantity InputScale 0 TimeArray Off
OrgName 11549 "[Report].[Query1.0].[Production cost]"
Origin Source Offset 1
Column "Production cost" Storage Float64 Scale 2
Size 1 Decimals 2
Class Quantity InputScale 0 TimeArray Off
Dimension 149 "Order date" DimType Date
EarliestDate 19010101 LatestDate 21001231 ManualPeriods False
DaysInWeek 127 NewCatsLock False ExcludeAutoPartitioning False
DimDefaultCategory 0 Categories Root 153 "Order date"
Inclusion Generate Lastuse 20070910 Date 0 Filtered False Suppressed
False
Sign False HideValue False IsKeyOrphanage False IsTruncated False
Blanks False
Drill 155 "By Order date" Inclusion Suppress
Filtered False Suppressed True PrimaryDrill True HideValue False
YearBegins 20070101 PartialWeek Split ExtraWeek None
WeekBegins Sunday Levels 161 "Year" Blanks "( blank )"
Inclusion Generate DateFunction Year Generate Need
RefreshLabel False RefreshDescription False RefreshShortName False
NewCatsLock False CatLabFormat "YYYY" Timerank 10
UniqueCategories True UniqueMove False Associations 163 "Order date"
AssociationType Type_Query AssociationRole Role_Source
AssociationReferenced "Order date" Associations 165 "Order date"
AssociationContext 155 AssociationType Type_Query
AssociationRole Role_OrderBy
AssociationReferenced "Order date"
SortOrder Int16 SortAs Ascending Levels 167 "Quarter" Blanks "(
blank )" Inclusion Generate
DateFunction Quarter Generate All
RefreshLabel False RefreshDescription False RefreshShortName False
NewCatsLock False
CatLabFormat 'YYYY "Q" Q' Timerank 20 UniqueCategories True UniqueMove
False
Associations 169 "Order date" AssociationType Type_Query AssociationRole
Role_Source
AssociationReferenced "Order date" Associations 171 "Order date"
AssociationContext 155 AssociationType Type_Query
AssociationRole Role_OrderBy
AssociationReferenced "Order date" SortOrder Int16 SortAs Ascending
Category 233 "20040101-20041231" Parent 155 Levels 161
OrderBy Drill 155 Value "2004" Label "2004" Lastuse 20070910
SourceValue "2004" Date 20040101 Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False
IsTruncated False Blanks False
Category 235 "20040101-20040331" Parent 233 Levels 167
OrderBy Drill 155 Value "20040101" Label "2004 Q 1" Lastuse 20070910
SourceValue "20040101" Date 20040101 Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False
IsTruncated False Blanks False
Category 243 "20040401-20040630" Parent 233 Levels 167
```

```
OrderBy Drill 155 Value "20040401" Label "2004 Q 2" Lastuse 20070910
SourceValue "20040401" Date 20040401 Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False
IsTruncated False  Blanks False
Category 251 "20040701-20040930" Parent 233 Levels 167
OrderBy Drill 155 Value "20040701" Label "2004 Q 3" Lastuse 20070910
SourceValue "20040701" Date 20040701 Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False
IsTruncated False Blanks False
Category 259 "20041001-20041231" Parent 233 Levels 167
OrderBy Drill 155 Value "20041001" Label "2004 Q 4" Lastuse 20070910
SourceValue "20041001" Date 20041001 Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False
IsTruncated False Blanks False
Category 581 "20050101-20051231" Parent 155 Levels 161
OrderBy Drill 155 Value "2005" Label "2005" Lastuse 20070910
SourceValue "2005" Date 20050101 Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False
IsTruncated False  Blanks False
Category 583 "20050101-20050331" Parent 581 Levels 167
OrderBy Drill 155 Value "20050101" Label "2005 Q 1" Lastuse 20070910
SourceValue "20050101" Date 20050101 Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False
IsTruncated False  Blanks False
Category 591 "20050401-20050630" Parent 581 Levels 167
OrderBy Drill 155 Value "20050401 " Label "2005 Q 2" Lastuse 20070910
SourceValue "20050401" Date 20050401 Filtered False Suppressed False
Sign False HideValueFalse IsKeyOrphanage False
IsTruncated False  Blanks False
Category 599 "20050701-20050930" Parent 581 Levels 167
OrderBy Drill 155 Value "20050701" Label "2005 Q 3" Lastuse 20070910
SourceValue "20050701" Date 20050701 Filtered False Suppressed False
Sign False HideValue FalseIsKeyOrphanage False
IsTruncated False  Blanks False
Category 607 "20051001-20051231" Parent 581 Levels 167
OrderBy Drill 155 Value "20051001" Label "2005 Q 4" Lastuse 20070910
SourceValue "20051001" Date 20051001 Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False
IsTruncated False Blanks False
Category 683 "20060101-20061231" Parent 155 Levels 161
OrderBy Drill 155 Value "2006" Label "2006" Lastuse 20070910
SourceValue "2006" Date 20060101 Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False
IsTruncated False Blanks False
Category 685 "20060101-20060331" Parent 683 Levels 167
OrderBy Drill 155 Value "20060101" Label "2006 Q 1" Lastuse 20070910
SourceValue "20060101"Date 20060101 Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False IsTruncated False
Blanks False
Category 693 "20060401-20060630" Parent 683 Levels 167
OrderBy Drill 155 Value "20060401" Label "2006 Q 2" Lastuse 20070910
SourceValue "20060401" Date 20060401 Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False
IsTruncated False Blanks False
Category 701 "20060701-20060930" Parent 683 Levels 167
OrderBy Drill 155 Value "20060701" Label "2006 Q 3" Lastuse 20070910
SourceValue "20060701" Date 20060701 Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False
IsTruncated False Blanks False
Category 709 "20061001-20061231" Parent 683 Levels 167
OrderBy Drill 155 Value "20061001" Label "2006 Q 4" Lastuse 20070910
SourceValue "20061001" Date 20061001 Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False
IsTruncated False Blanks False
MapDrills
MapDrill 155
ViewName 157 "All Categories" Type All ViewCustomView 0
```

```
ViewName 159 "Omit Dimension" Type Omit ViewCustomView 0 Dimension
195
"Order method" DimType Regular NewCatsLock False ExcludeAutoPartitioning
False
DimDefaultCategory 0 Categories Root 197 "Order method" Inclusion
Generate
Lastuse 20070910 Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False
IsTruncated False Blanks False
Drill 199 "By Order method" Inclusion Suppress Filtered False Suppressed
True
PrimaryDrill True HideValue False Levels 205
"Order method" Blanks "( Blank )" DateFunction None Generate None
RefreshLabel False RefreshDescription False
 RefreshShortName False NewCatsLock False Timerank 0 UniqueCategories
False
 UniqueMove False Associations 207
"Order method code" AssociationType Type_Query AssociationRole Role_Source
AssociationReferenced "Order method code"
Associations 209 "Order method" AssociationType Type_Query AssociationRole
Role_Label
AssociationReferenced "Order method"
Category 267 "7" Parent 199 Levels 205 Label "Sales visit" Lastuse
20070910
SourceValue "7" Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False IsTruncated False
Blanks False
Category 285 "4" Parent 199 Levels 205
Label "E-mail" Lastuse 20070910
SourceValue "4" Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False IsTruncated False
Blanks False
Category 309 "5" Parent 199 Levels 205 Label "Web" Lastuse 20070910
SourceValue "5" Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False IsTruncated False
Blanks False
Category 319 "2" Parent 199 Levels 205 Label "Telephone" Lastuse
20070910
SourceValue "2" Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False IsTruncated False
Blanks False
Category 357 "1" Parent 199 Levels 205 Label "Fax" Lastuse 20070910
SourceValue "1" Filtered False Suppressed False
Sign FalseHideValue False IsKeyOrphanage False IsTruncated False
Blanks False
Category 437 "3" Parent 199 Levels 205 Label "Mail" Lastuse 20070910
SourceValue "3" Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False IsTruncated False
Blanks False
Category 475 "8" Parent 199 Levels 205 Label "Special" Lastuse 20070910
SourceValue "8" Current Filtered False Suppressed False
Sign False HideValue False IsKeyOrphanage False IsTruncated False
Blanks False
MapDrills
MapDrill 199
ViewName 201 "All Categories" Type All
ViewCustomView 0 ViewName 203 "Omit Dimension" Type Omit
ViewCustomView 0 ViewName 11555 "Authors~User View" ViewCustomView
11529
Apex 197 Filter 319
ViewName 11557 "Sub Authors~User View"
ViewCustomView 11537 Apex 197 Filter 319 Measure 225 "Quantity"
Rollup Sum
IgnoreMissingValue False Storage Float64 OutPutScale 0 Decimals
0 ReverseSign False
IsCurrency False IsFolder False DrillThrough False EndList
Associations 227 "Quantity" AssociationType Type_Query AssociationRole
```

```
Role_Source
AssociationReferenced "Quantity" CustomView 11529 "Authors" DimensionView
149 "All Categories"
DimensionView 195 "Authors~User View" MeasureInclude 225 Yes CustomView
11537 "Sub Authors"
DimensionView 149 "All Categories" DimensionView 195 "Sub Authors~User
View" MeasureInclude 225 Yes
CustomViewChildList 11529 StartList 11537 EndList
CustomViewChildList 11537 StartList EndList
SecurityNameSpace 11533 "Cognos" SecurityNameSpaceCAMID 'CAMID(":")'
SecurityObject
11531
'CAMID(":Authors")' SecurityObjectDisplayName "Authors"SecurityObjectType
SecurityType_Role
CustomViewList 11529 EndList
SecurityObject 11539 'CAMID(":Analysis Users")' SecurityObjectDisplayName
"Analysis
Users"
SecurityObjectType SecurityType_Role CustomViewList 11537 EndList
SecurityNameSpace 11563 "GOnamespace" SecurityNameSpaceCAMID 'CAMID
("GOnamespace")'
SecurityObject 11561 'CAMID("GOnamespace:r:authid=2589996611")'
SecurityObjectDisplayName "Root User Class"
SecurityObjectType SecurityType_Role CustomViewList 11537 EndList
Cube 11535 "Cube1" EncryptedPW
"DD32E203AA9AFC5C26233A515A4E83A1DD32E203AA9AFC5C26233A515A4E83A1DD32E20"
"3AA9AFC5C26233A515A4E83A1DD32E203AA9AFC5C26233A515A4E83A1DD32E203AA9AFC"
"5C26233A515A4E83A1DD32E203AA9AFC5C26233A515A4E83A1DD32E203AA9AFC5C26233"
"A515A4E83A1DD32E203AA9AFC5C26233A515A4E83A1DD32E203AA9AFC5C26233A515A4E"
"83A1DD32E203AA9AFC5C26233A515A4E83A1DD32E203AA9AFC5C26233A515A4E83A1DD3"
"2E203AA9AFC5C26233A515A4E83A1DD32E203AA9AFC5C26233A515A4E83A1DD32E203AA"
"9AFC5C26233A515A4E83A1DD32E203AA9AFC5C26233A515A4E83A1DD32E203AA9AFC5C2"
"6233A515A4E83" Status New CubeCreation On Optimize Default ConsolidatedRecords
10000000
PartitionSize 500000 PassesNumber 5 Compress False DatabaseInfo
"Local;"
IncrementalUpdate False ServerCube False CubeStamp 0 CubeCycle 0
BlockParentTotals False Caching False UseAlternateFileName False
DrillThrough False EndList DataSourceSignon False PublisheEnable
True PublishStatus None
DimensionView 149 "All Categories" DimensionView 195 "All Categories"
MeasureInclude
225 Yes
PowerCubeCustomViewList 11537 EndList
AllocationAdd Measure 225 Type Default
```

Index

D

N

O

P

Q

R

S

T

U

V

W

Y